



TOPS TECHNOLOGIES

Training | Outsourcing | Placement | Study Abroad

Diploma in Native Application

Software Engineering Course

Content

Module 1 - [Overview of IT Industry]

Module 2 - [Fundamentals of Programming]

Module 3 - [OOP Concept]

Module 4 - [HTML & CSS]

Module 5 - [Database]

Module - 1

[Overview of IT Industry]

What is Program

- Program- It is a set of Instructions

Ex. 1- Instructions to your

Pet Ex. 2- Starting your

Computer



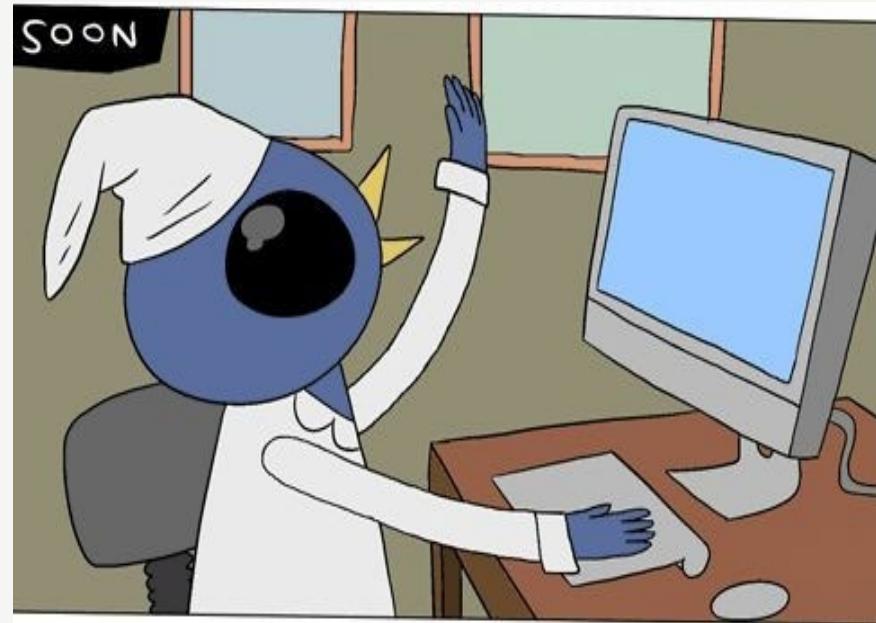
```
>Hello world!  
>_
```

A black terminal window with white text. It displays two lines of code: '>Hello world!' on the first line and '>_' on the second line.

What is Programming

- Programming- To create a Program.

Ex. 1- Using Keyboard & Mouse



Types of Programming Language

- Procedural Programming
 - Ex.- C Language



Types of Programming Language

- Object Oriented Programming
 - Ex.- C++ Language



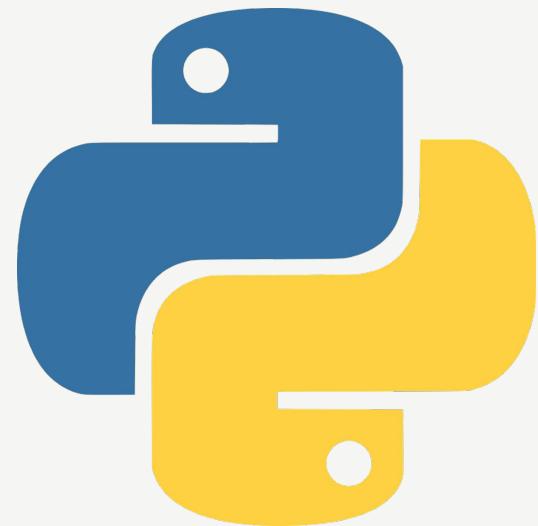
Types of Programming Language

- Logical Programming
 - Ex.- Prolog Language



Types of Programming Language

- Functional Programming
 - Ex.- Python Language



World Wide Web - WWW

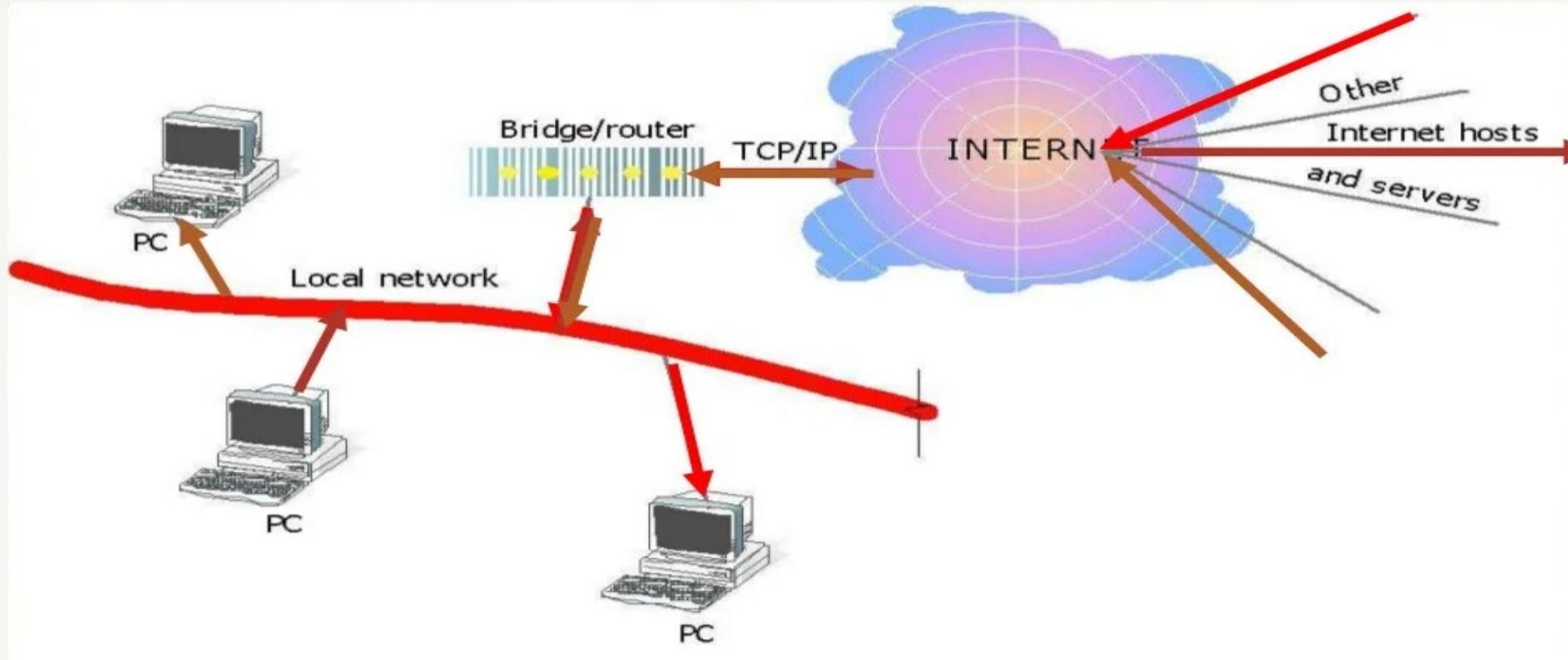
- known as a Web
- is a collection of websites or web pages stored in web servers
- connected to local computers through the internet
- These websites contain text pages, digital images, audios, videos, etc.
- Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc.
- The WWW, along with internet, enables the retrieval and display of text and media to your device.



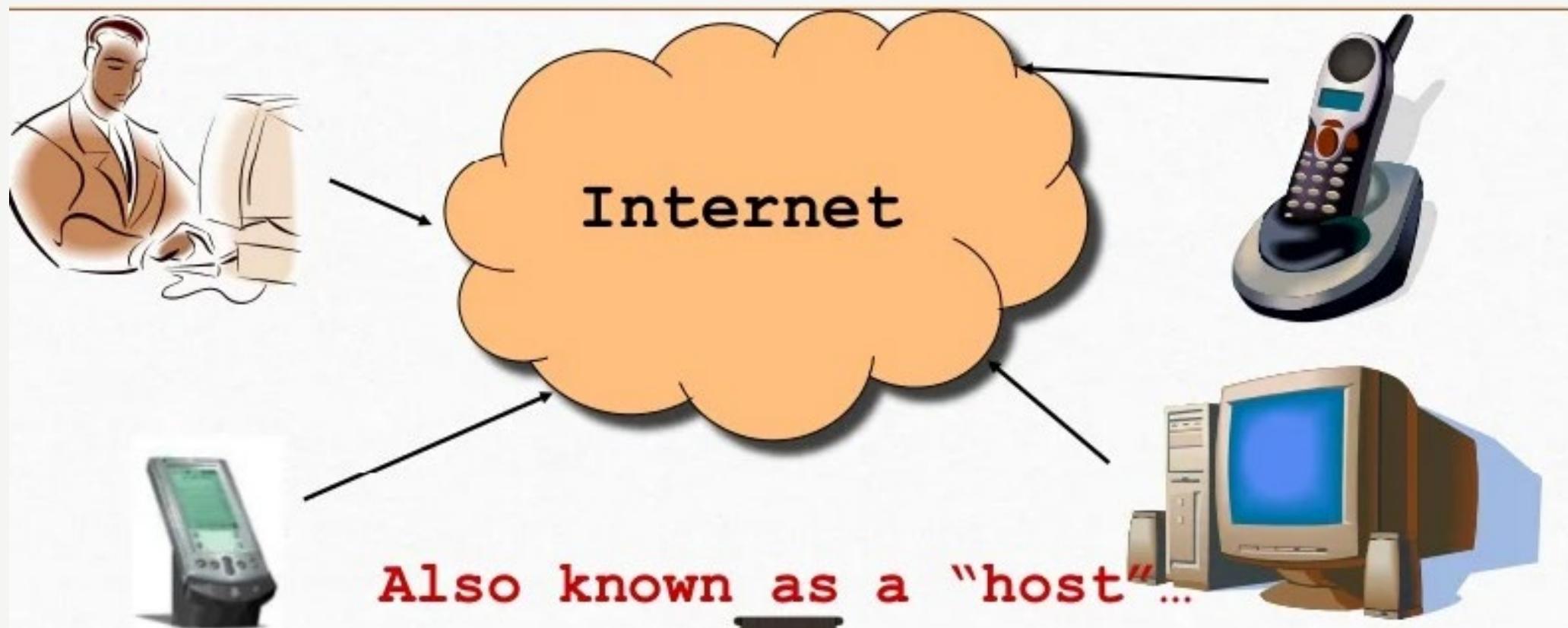
WWW – How Internet Works?

- A server is where websites are stored, and it works a lot like your computer's hard drive.
- Once the request arrives, the server retrieves the website and sends the correct data back to your computer.
- One of the best features of the Internet is the ability to communicate almost instantly with anyone in the world.
- **Email** is one of the oldest and most universal ways to communicate and share information on the Internet, and billions of people use it. **Social media** allows people to connect in a variety of ways and build communities online.

WWW – How Internet Works?



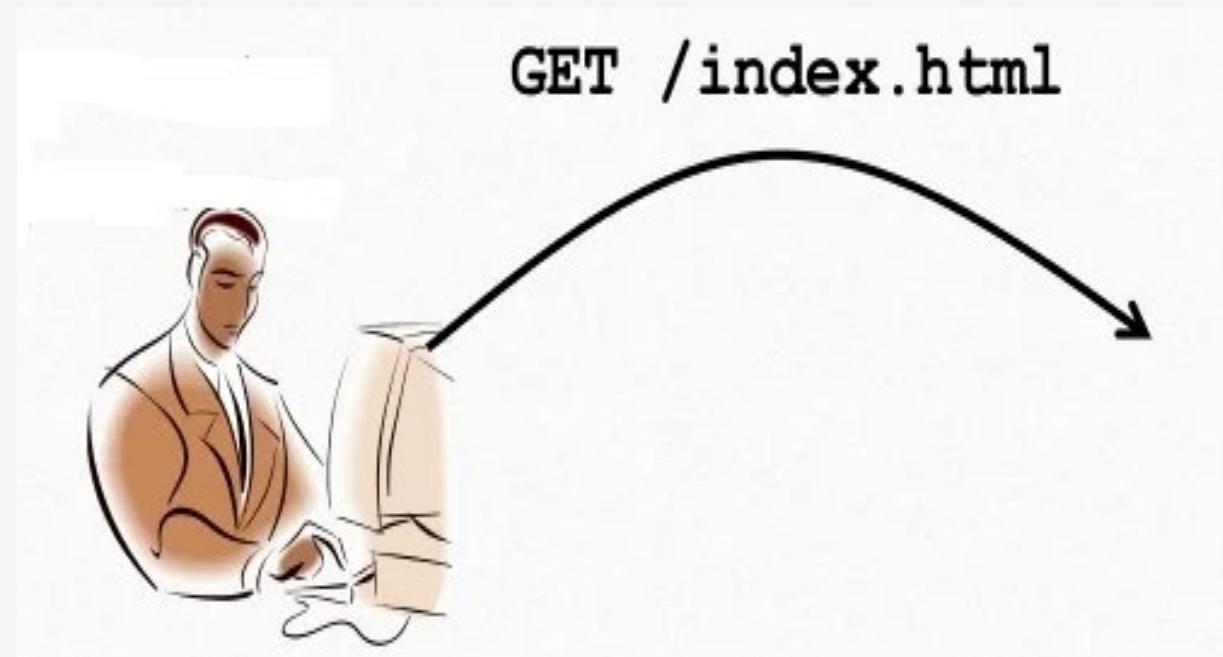
Network layers on Client & Server



Client & Servers

1. Client Program

- Running on end host
- Requests Services
- E.g. Web Browser



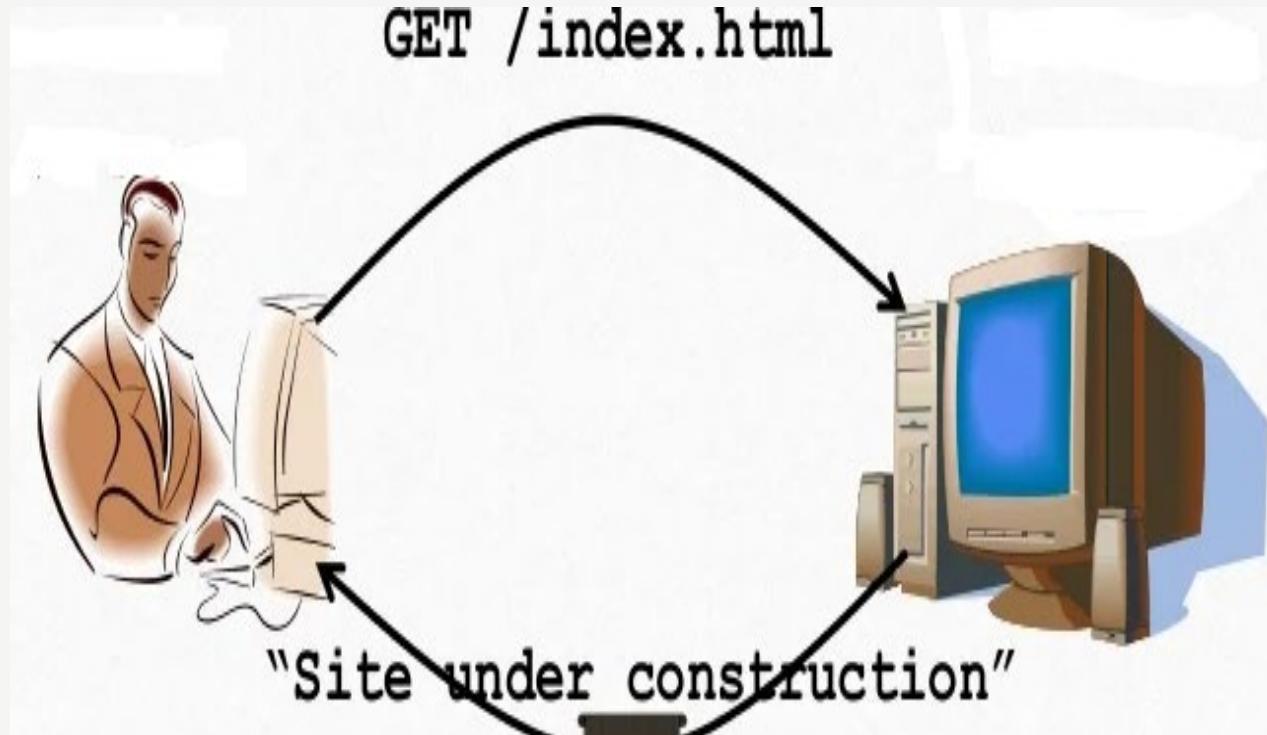
Client & Servers

1. Client Program

- Running on end host
- Requests Services
- E.g. Web Browser

2. Server Program

- Running on end host
- Provides Services
- E.g. Webserver



Client – Servers Communication

1. Client “sometimes on”

- Initiates a request to the server when interested
- E.g. Web Browser on your laptop or cell phone
- Doesn't communicate directly with other clients
- Needs to know the server's address

2. Server is “always on”

- Services requests from many client hosts
- E.g. Web Server for the www.example.com web site
- Doesn't initiate contact with the clients
- Needs a fixed, well-known address

Types of Internet connection

1. Digital subscriber line(DSL)
2. Cable Internet
3. Fiber Optic
4. Satellite Internet
5. Wireless
6. Broadband over Power lines(BPL)

Major Applications of the Internet

- 4 major applications of internet, which are given below.

1. Social Media Internet Applications

e.g .Facebook,Instagram,Twitter,LinkedIn...

2. Communications Internet Applications

e.g. Email,Skype,Zoom,Whatsapp...

3. Entertainment Internet Applications

e.g. Netflix,Hotstar,Youtube, Amazon prime video...

4. Travel Internet Applications

e.g Google Trips,Google Map,trivago...

What are protocols?

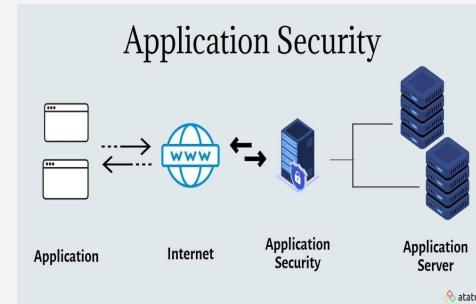
- A Network Protocol is a group of rules accompanied by the network.
- Network protocols will be formalized requirements and plans composed of rules, procedures, and types that describe communication among a couple of devices over the network.
- The protocol can be described as an approach to rules that enable a couple of entities of a communication program to transfer information through any type of variety of a physical medium.
- The protocol identifies the rules, syntax, semantics, and synchronization of communication and feasible error managing methods. In this article, we will discuss the different types of networking protocols.

Types of Protocols

1. HTTP or HTTPS
2. FTP(File Transfer protocols)
3. Email Protocols(POP3,SMTP)
4. TCP(Transmission control protocol) and UDP(User Datagram Protocol)

What is Application Security?

- Application security refers to security precautions used at the application level to prevent the theft or hijacking of data or code within the application.
- It includes security concerns made during application development and design, as well as methods and procedures for protecting applications once they've been deployed.
- All tasks that introduce a secure software development life cycle to development teams are included in application security shortly known as AppSec.
- Its ultimate purpose is to improve security practices and, as a result, detect, repair, and, ideally, avoid security flaws in applications.
- It covers the entire application life cycle, including requirements analysis, design, implementation, testing, and maintenance..



What is Application Security?

- All tasks that introduce a secure software development life cycle to development teams are included in application security shortly known as AppSec.
- Its ultimate purpose is to improve security practices and, as a result, detect, repair, and, ideally, avoid security flaws in applications.
- It covers the entire application life cycle, including requirements analysis, design, implementation, testing, and maintenance..

Software Applications

What is Application Software :

- It is a type of software application that helps in the automation of the task based on the Users Input.
- It can perform single or multiple tasks at the same period of time.
- There are the different application which helps us in our daily life to process our instructions based on certain rules and regulations.
- Application Software helps in providing a graphical user interface to the user to operate the computer for different functionality.
- The user may use the computer for browsing the internet, accessing to email service, attending meetings, and playing games.
- Different high-level languages are used to build application software.

Types of Application Software

- Application software
- System software
- Driver software
- Middleware
- Programming software

Application Software

- The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application.
- An application can be self-contained, or it can be a group of programs that run the application for the user.
- Examples of Modern Applications include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.

Example:Microsoft Office, Paint, Powerpoint etc..

System Software

- These software programs are designed to run a computer's application programs and hardware.
- System software coordinates the activities and functions of the hardware and software.
- It controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in.
- The OS is the best example of system software; it manages all the other computer programs.
- Other examples of system software include the firmware, computer language translators and system utilities..

Example:Notepad ,Calculator etc..

Driver Software

- Also known as device drivers, this software is often considered a type of system software.
- Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks.
- Every device that is connected to a computer needs at least one device driver to function.
- Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones and printers.

Example: Audio Driver, Video Driver etc..

Middleware

- The term *middleware* describes software that mediates between application and system software or between two different kinds of application software.
For example, middleware enables Microsoft Windows to talk to Excel and Word.
- It is also used to send a remote work request from an application in a computer that has one kind of OS, to an application in a computer with a different OS. It also enables newer applications to work with legacy ones.

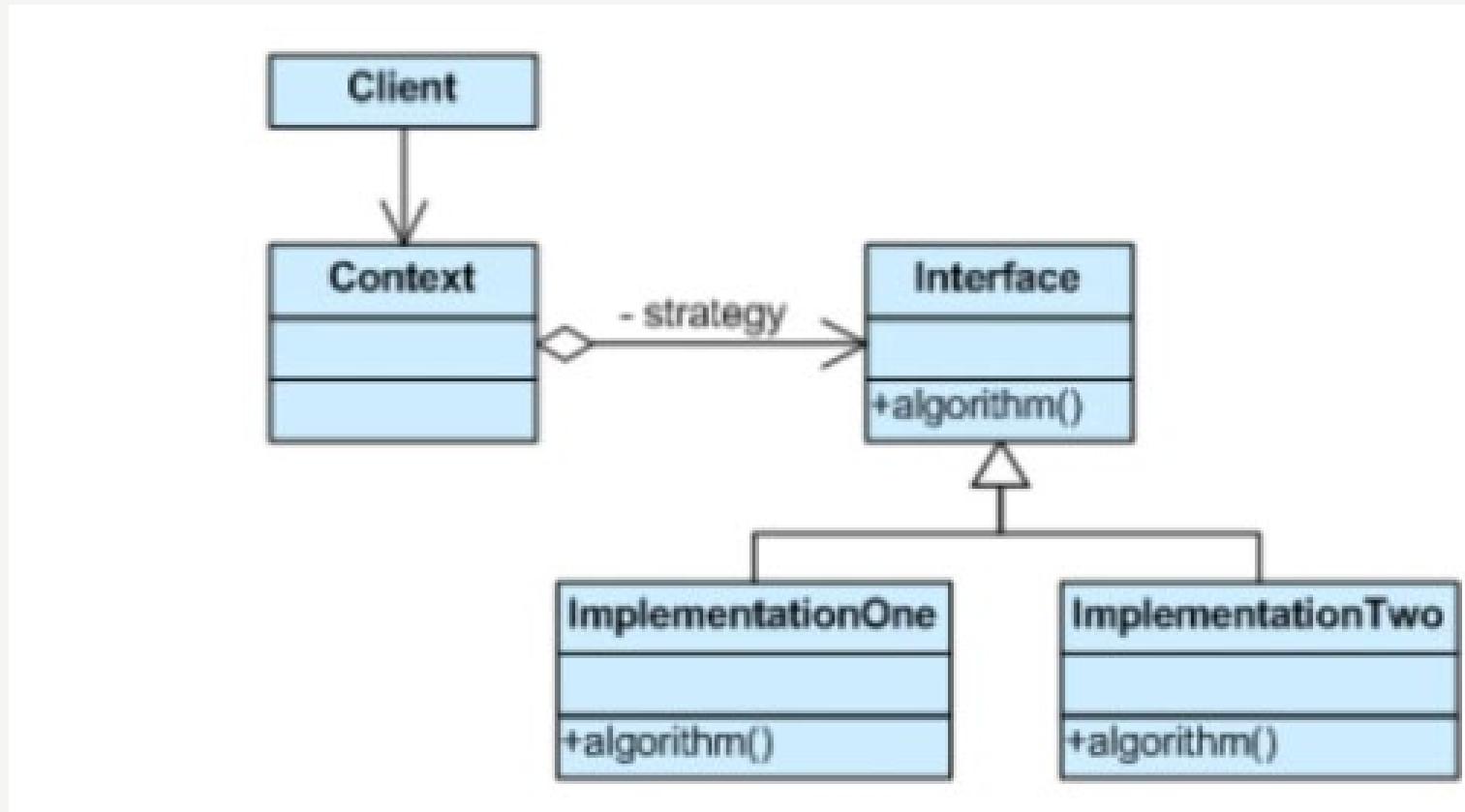
Example: database middleware, application server middleware

Programming Software

- Computer programmers use programming software to write code.
Programming software and programming tools enable developers to develop, write, test and debug other software programs.
- Examples of programming software include assemblers, compilers, debuggers and interpreters.

Examples : Turbo c,Eclipse,Sublime etc..

Architecture



Application and Examples

- Shopping mall Example
 - 1. Accept Customer Details
 - 2. calculate the bill
 - 3. Apply Discount Based on day of the week
 - (a) Monday Low discount -10%
 - (b) Thursday High discount-50%



- Shopping Mall

- 1. Accept customer detail
- 2. Calculate bill amount
- 3. Apply discount based on day of week
 - 1. Monday - Low discount - 10%
 - 2. Thursday - High discount - 50%

- Customer Detail
- CalculateBill
- GetFinalBill

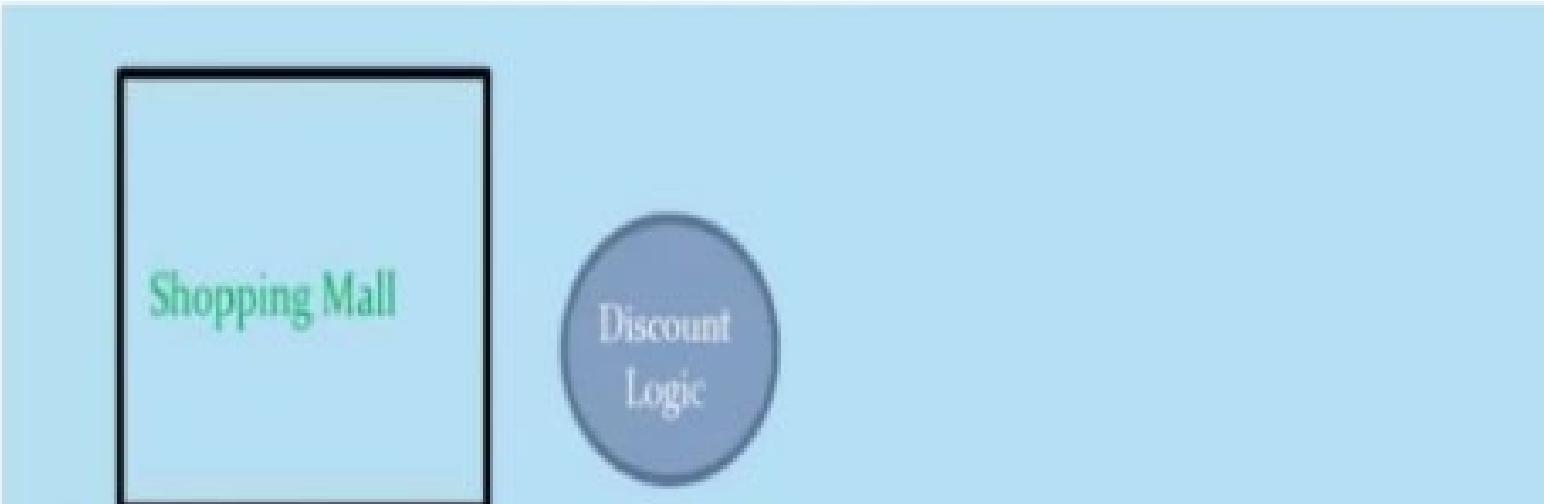
Shopping Mall will contain discount logic.

Shopping Mall

Discount
Logic

Open closed principle – software entities should be open for extension,
but closed for modification.

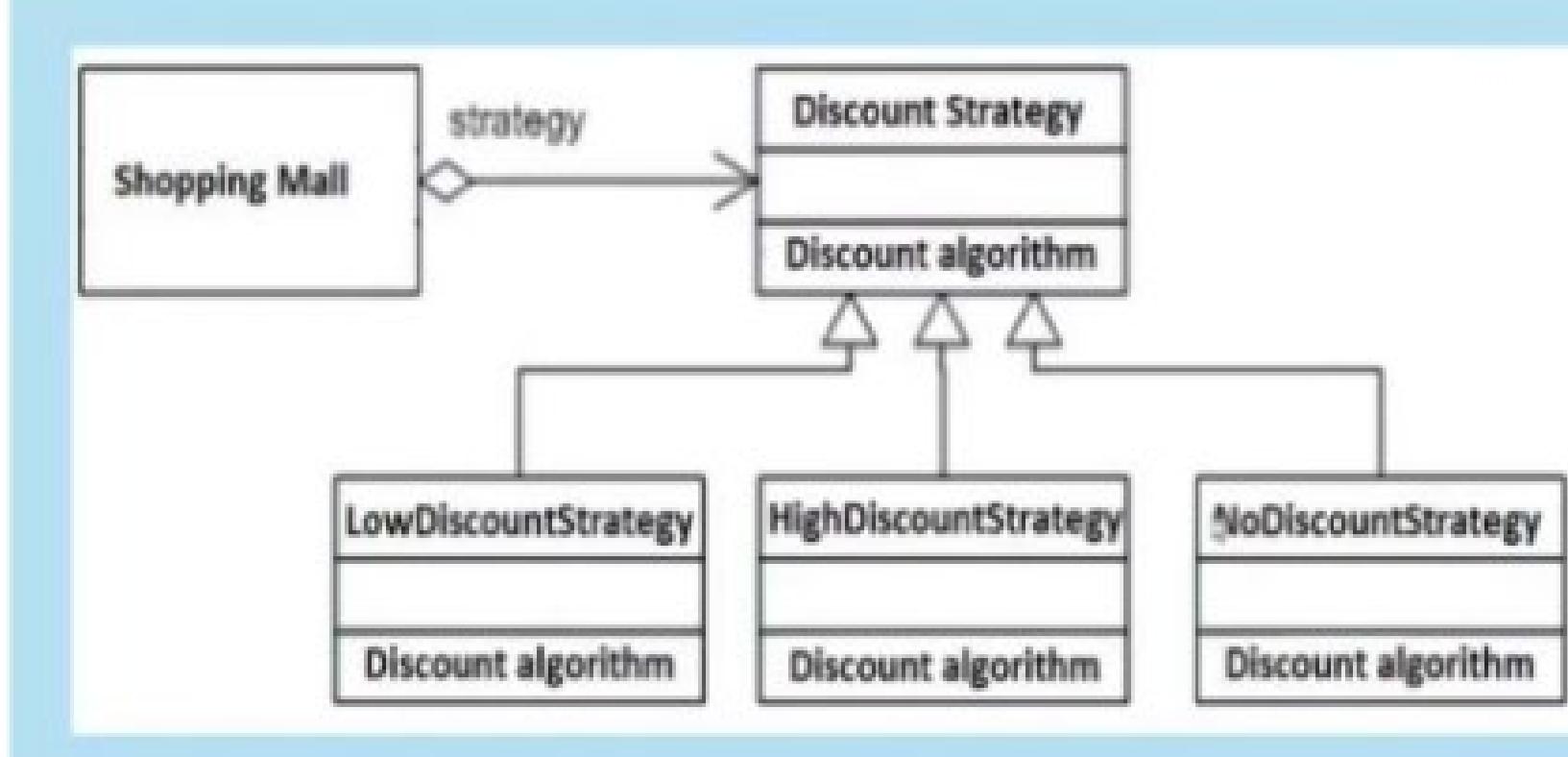
New discount strategy may be applied in future



Shopping Mall

Discount
Logic

Open closed principle – software entities should be open for extension,
but closed for modification.



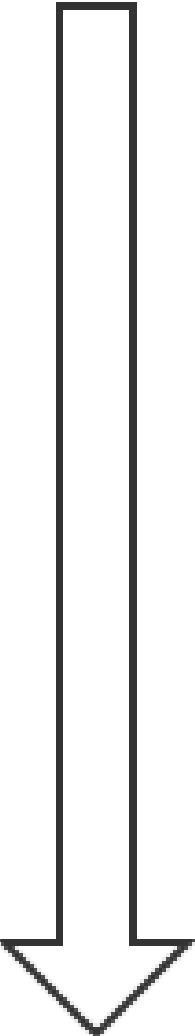
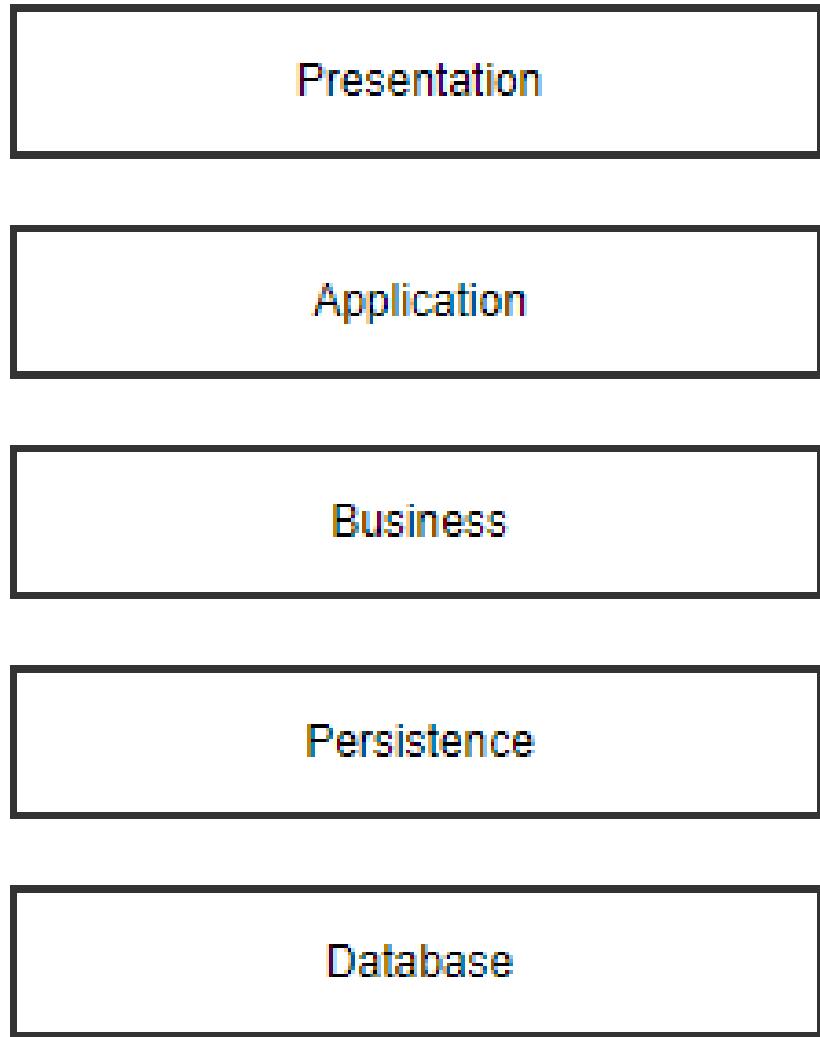
Software Architecture

- Software architecture is the blueprint of building software. It shows the overall structure of the software, the collection of components in it, and how they interact with one another while hiding the implementation.
- This helps the software development team to clearly communicate how the software is going to be built as per the requirements of customers.
- There are various ways to organize the components in software architecture. And the different predefined organization of components in software architectures are known as software architecture patterns.

- A lot of patterns were tried and tested. Most of them have successfully solved various problems. In each pattern, the components are organized differently for solving a specific problem in software architectures.
- Well, I hope you don't want to bore yourself by reading the endless types of software architecture patterns.

Layers in Software Architecture

- 1. Presentation layer**
- 2. Application layer**
- 3. Business layer**
- 4. Persistence layer**
- 5. Database layer**



Calls between layers flow downwards

1. Presentation layer

The presentation layer, also called the UI layer, handles the interactions that users have with the software. It's the most visible layer and defines the application's overall look and presentation to the end-users. This is the tier that's most accessible, which anyone can use from their client device, like a desktop, laptop, mobile phone or tablet.

2. Application layer

The application layer handles the main programs of the architecture. It includes the code definitions and most basic functions of the developed application. This is the layer that programmers spend most of their time in when working on the software. You can use this layer to implement specific coordination logic that doesn't align exactly with either the presentation or business layer.

3. Business layer

The business layer, also called the domain layer, is where the application's business logic operates. Business logic is a collection of rules that tell the system how to run an application, based on the organization's guidelines. This layer essentially determines the behavior of the entire application. After one action finishes, it tells the application what to do next.

4. Persistence layer

The persistence layer, also called the data access layer, acts as a protective layer. It contains the code that's necessary to access the database layer. This layer also holds the set of codes that allow you to manipulate various aspects of the database, such as connection details and SQL statements.

5. Database layer

The database layer is where the system stores all the data. It's the lowest tier in the software architecture and houses not only data but indexes and tables as well. Search, insert, update and delete operations occur here frequently. Application data can store in a file server or database server, revealing crucial data but keeping data storage and retrieval procedures hidden.

Environments in industry

There are different Types of environments in Industry.

1. The analysis and design environment
2. The development environment
3. The common build environment
4. The testing environment
5. The production environment



Environments in industry

1. Analysis & Design Environment

- The analysis and design environment is aligned to the **planning and analysis phases** of the SDLC. In this environment, the main processes that take place include carrying out an in-depth examination of the current system and the proposed system. The system architecture is also defined and includes developing the design of the hardware, software, and network requirements for the system. Within this environment, systems and business analysts work closely with software engineers.

2. The development environment

- The development environment can also be a physical space where development takes place and where software engineers interact. Another example of the development environment is the **integrated development environment** (IDE). The IDE provides a platform where tools and development processes are coordinated in order to provide software engineers a convenient way of accessing the resources they require during the development process.

Environments in industry

3. The common build environment

- The **common build environment** is closely aligned to the development phase of the SDLC. In this environment, software engineers merge the work done in the development environment. Within this environment, software engineers build systems. These are used to automate the process of software compilation.

4. The testing environment

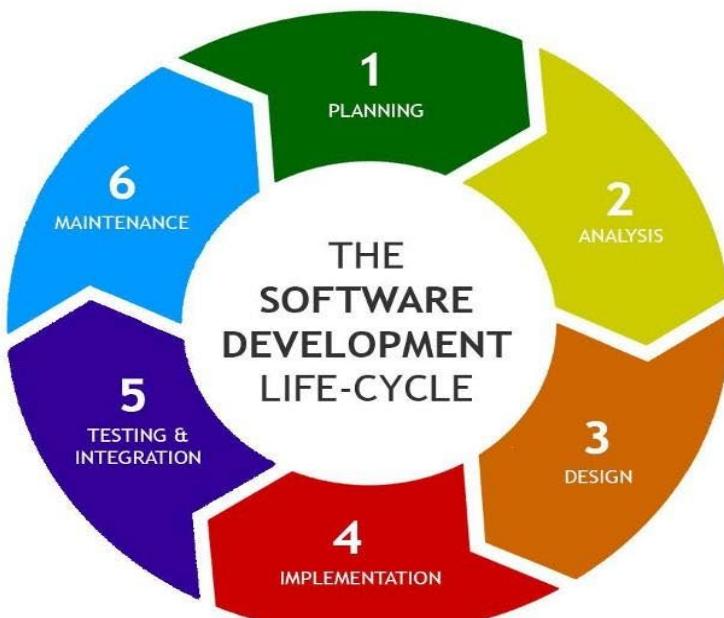
- The test environment is where testing teams evaluate the application/quality. program's This also allows computer programmers to find out and solve any defects that may interfere with the application's smooth operation or degrade the user experience.

5. The production environment

- When the end-user use a web/mobile application, the program is operating on a production server. It's been created in the production environment.

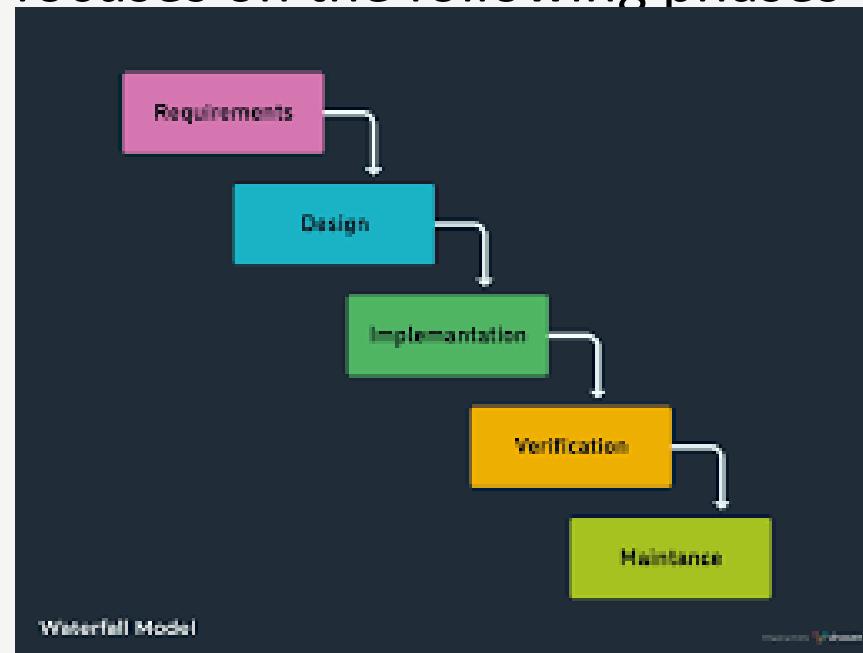
SDLC

- The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software.



SDLC Methodology

- The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. In detail, the SDLC methodology focuses on the following phases of software development:
 - Requirement Gathering
 - Analysis
 - Designing
 - Implementation
 - Testing
 - Maintenance



Programming

There are countless definitions of what computer programming is, but here is Ours.

“Programming is how you get computers to solve problems.”

There are two key phrases here that are important:

- **You:** without the programmer (you), the computer is useless. It does what **you** tell it to do.
- **Solve problems:** computers are tools. They are complex tools, admittedly, but they are not mysterious or magical: they exist to make tasks easier.

Program m ing

Computer programs make computers work

Computer programs (or software) are what makes computers work. Without software, modern computers are just complicated machines for turning electricity into heat. It's software on your computer that runs your operating system, browser, email, games, movie player – just about everything.

Program m ing

Programming is creative:

Programming is a creative task: there is no right or wrong way to solve a problem, in the same way, that there is no right or wrong way to paint a picture.

There are choices to be made, and one way may seem better than another, but that doesn't mean the other is wrong! With the right skills and experience, a programmer can craft software to solve an unlimited number of problems – from telling you when your next train will arrive at playing your favourite music.

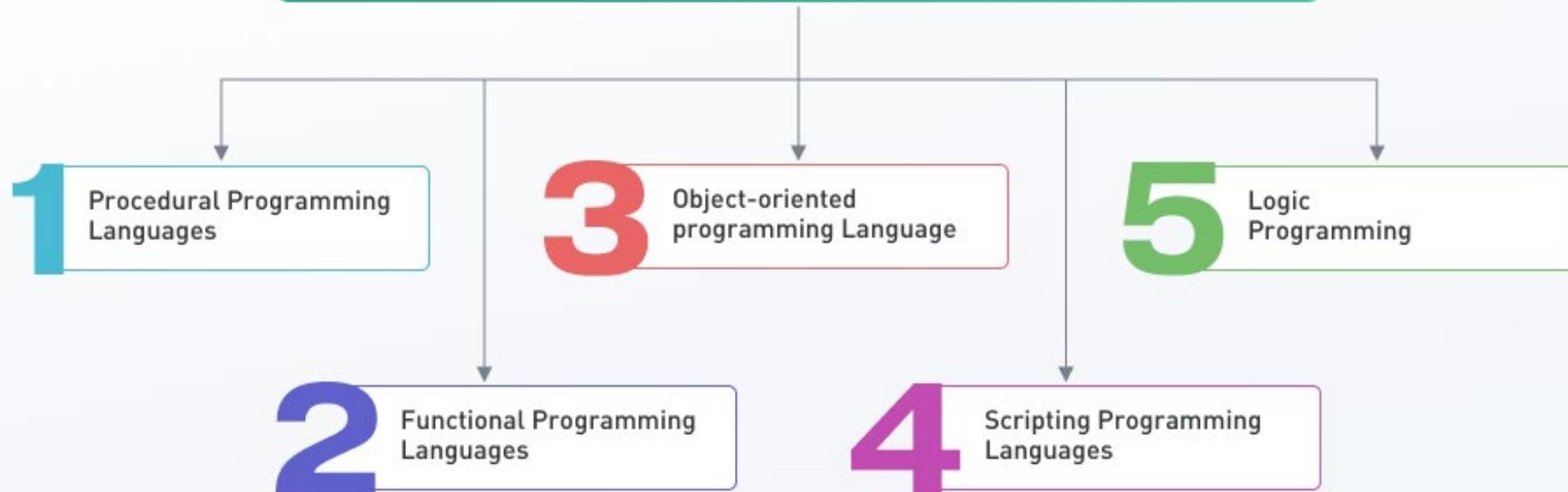
The possibilities are constrained only by your imagination. That's why I love programming. Programming having a multiple Programming Languages.

Programming Languages

- A **programming language** is a computer language programmers use to develop software programs, scripts, or other sets of instructions for computers to execute.
- Although many languages share similarities, each has its own syntax. Once a programmer learns the language's rules, syntax, and structure, they write the source code in a text editor or IDE. Then, the programmer often compiles the code into machine language that can be understood by the computer. Scripting languages, which do not require a compiler, use an interpreter to execute the script.
- A programming language is a set of instructions that can be used to interact with and control a computer. These languages are used to design websites, create apps, develop operating systems, control spacecraft, and analyze data. Programming languages are necessary because computers can't understand English. Programming languages bridge this gap by helping programmers translate their commands into something that the computer can understand and execute.

Types of Programming Languages

TYPES OF PROGRAMMING



Types Of Programming Languages

1. Low-level programming language

Low-level language is **machine-dependent (0s and 1s)** programming language. The processor runs low- level programs directly without the need of a compiler or interpreter, so the programs written in low-level language can be run very fast.

Low-level language is further divided into two parts -

i. Machine Language

Machine language is a type of low-level programming language. It is also called as **machine code or object code**. Machine language is easier to read because it is normally displayed in binary or hexadecimal form (base 16) form. It does not require a translator to convert the programs because computers directly understand the machine language programs.

The advantage of machine language is that it helps the programmer to execute the programs faster than the high-level programming language.

ii. Assembly Language

Assembly language (ASM) is also a type of low-level programming language that is designed for specific processors. It represents the set of instructions in a **symbolic and human-understandable form**. It uses an assembler to convert the assembly language to machine language.

The advantage of assembly language is that it requires less memory and less execution time to execute a program.

2. High-level programming language

High-level programming language (HLL) is designed for **developing user-friendly software programs and websites**. This programming language requires a compiler or interpreter to translate the program into machine language (execute the program).

The main advantage of a high-level language is that it is **easy to read, write, and maintain**.

High-level programming language includes **Python, Java, JavaScript, PHP, C#, C++, Objective C, Cobol, Perl, Pascal, LISP, FORTRAN, and Swift** programming language.

A high-level language is further divided into three parts -

i. Procedural Oriented programming language

Procedural Oriented Programming (POP) language is derived from structured programming and based upon the procedure call concept. It divides a program into small procedures called **routines or functions**.

Procedural Oriented programming language is used by a software programmer to create a program that can be accomplished by using a programming editor like IDE, Adobe Dreamweaver, or Microsoft Visual Studio.

The advantage of POP language is that it helps programmers to easily track the program flow and code can be reused in different parts of the program.

Example: C, FORTRAN, Basic, Pascal, etc.

ii. Object-Oriented Programming language

Object-Oriented Programming (OOP) language is **based upon the objects**. In this **programming language**, programs are divided into small parts **called objects**. It is used to implement real-world entities like inheritance, polymorphism, abstraction, etc in the program to makes the program reusable, efficient, and easy-to-use.

The main advantage of object-oriented programming is that OOP is faster and easier to execute, maintain, modify, as well as debug.

Example: C++, Java, Python, C#, etc.

iii. Natural language

Natural language is a **part of human languages** such as English, Russian, German, and Japanese. It is used by machines to understand, manipulate, and interpret human's language. It is used by developers to **perform tasks such as translation, automatic summarization, Named Entity Recognition (NER), relationship extraction, and topic segmentation.**

The main advantage of natural language is that it helps users to ask questions in any subject and directly respond within seconds.

3. Middle-level programming language

Middle-level programming language **lies between the low-level programming language and high-level programming language.** It is also known as the intermediate programming language and pseudo-language.

A middle-level programming language's advantages are that it supports the features of high-level programming, it is a user-friendly language, and closely related to machine language and human language.

Example: C, C++, language

Important Programming Language

- Development is Mainly Divided into two Parts

Front end Development

HTML

CSS

JAVASCRIPT

React

Angular

Important Programming Language

Backend Development

In Backend you have a many languages you can choose anyone language

If you are Choosing web development platform then you have a Mainly Three Languages

PHP

JAVA

PYTHON

Dot Net

Important Programming Language

HTML:

HTML—“HyperText Markup Language”—is **the language used to tell your web browser what each part of a website is**. So, using HTML, you can define headers, paragraphs, links, images, and more, so your browser knows how to structure the web page you're looking at.

CSS:

CSS makes the front-end of a website shine and it creates a great user experience. Without CSS, websites would be less pleasing to the eye and likely much harder to navigate. In addition to layout and format, CSS is responsible for font color and more.

Important Programming Language

JAVASCRIPT:

JavaScript has evolved over the past 25 years **to become a versatile and accessible programming language for working with web browsers.** Developers use JavaScript to build complex interactive websites and browser games, and to connect servers to websites and web applications

Important Programming Language

Programming Language in Backend:

The backend (or “server-side”) is the portion of the website you don't see. It's **responsible for storing and organizing data, and ensuring everything on the client-side actually works.** The backend communicates with the frontend, sending and receiving information to be displayed as a web page.

Writing Source Code

What is Source Code?

Source code is the source of a computer program. It contains declarations, instructions, functions, loops and other statements, which act as instructions for the program on how to function. Programs may contain one or more source code text files, which can be stored on a computer's hard disk, in a database, or be printed in books of code snippets.

Important Notes For Source Code

- Coding Must Have Comments
- Coding Must Have a Proper Architecture
- Coding Must Have a
- Declaring Coding Versions
- Divide your code in packages

Running Code

When you will run your code two things are Most Important

1.Compiler

2.Interpreter

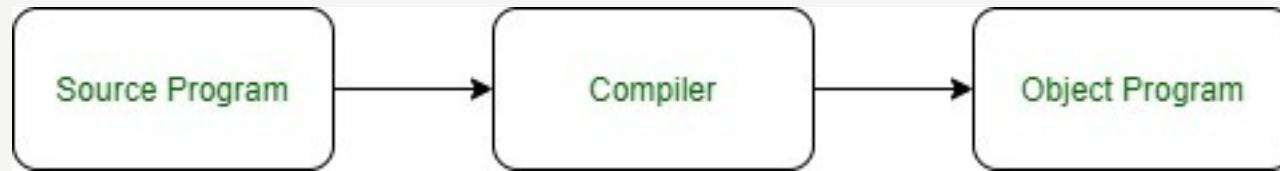
1.Compiler:

It is a translator which takes input i.e., High-Level Language, and produces an output of low-level language i.e. machine or assembly language.

- A compiler is more intelligent than an assembler it checks all kinds of limits, ranges, errors, etc.

- But its program run time is more and occupies a larger part of memory.

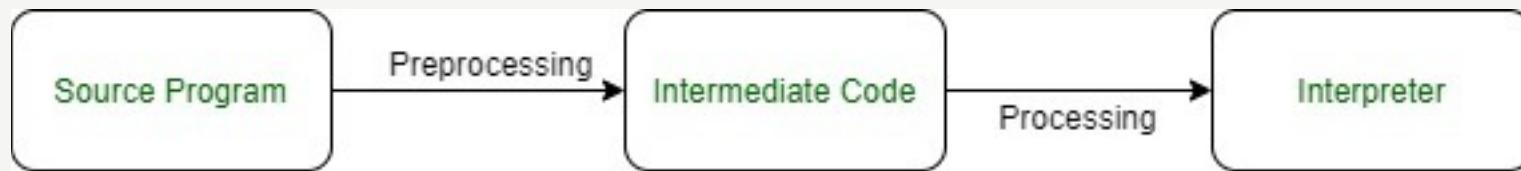
It has a slow speed because a compiler goes through the entire program and then translates the entire program into machine codes.



2. Interpreter :

An interpreter is a program that translates a programming language into a comprehensible language. –

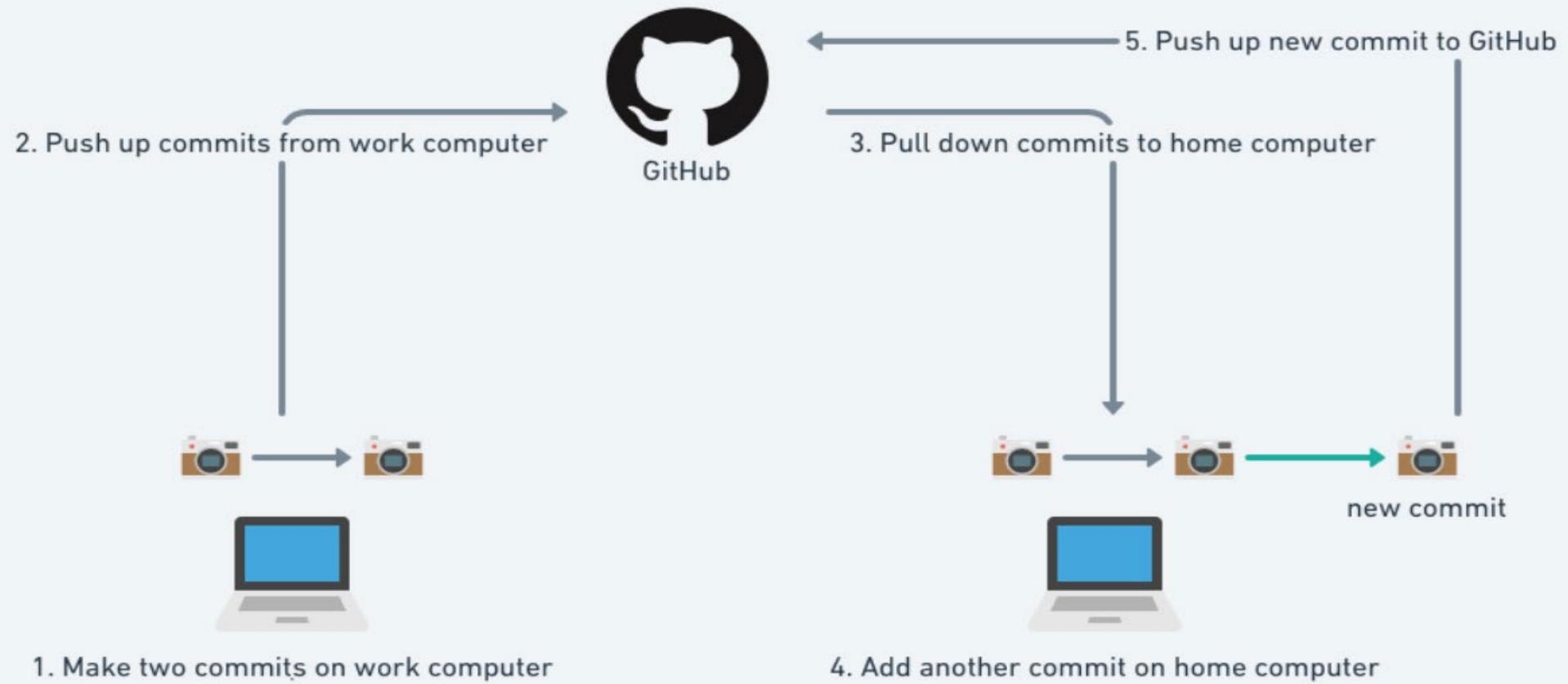
- It translates only one statement of the program at a time.
- Interpreters, more often than not are smaller than compilers.



GitHub

- GitHub is one of the most popular resources for developers to share code and work on projects together. It's free, easy to use, and has become central in the movement toward open-source software.
- Git is used for managing the changes to a project over time. A project might be just a single file, a handful of files, or thousands of files. Those files can be anything from plain text to images or videos.
- Because Git is focused on managing changes, it is often used as a collaboration tool allowing people to work on the same project at the same time. By tracking their individual changes, Git can bring everything together to the final version.

Working with GitHub



Module - 2

[Fundamentals of Programing]

Basic Concepts of Programming Languages

1. Syntax
2. Data Structures
3. Variables
4. Operators
5. Control & Looping Structures
6. Functions
7. Arrays & Strings
8. File Handling

1. Basic Syntax

Syntax of Programming Languages

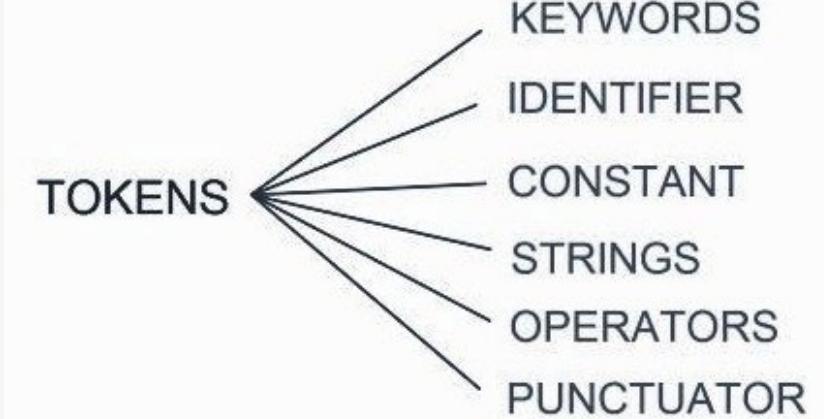
1. Syntax

- Syntax refers to the rules that define the structure of a language. Syntax in computer programming means the rules that control the structure of the symbols, punctuation, and words of a programming language.
- Without syntax, the meaning or semantics of a language is nearly impossible to understand.
- For example, a series of English words, such as — subject a need and does sentence a verb — has little meaning without syntax.
- If the syntax of a language is not followed, the code will not be understood by a compiler or interpreter.
- **Compilers** convert programming languages like Java or C++ into binary code that computers can understand. If the syntax is incorrect, the code will not compile.
- **Interpreters** execute programming languages such as JavaScript or Python at runtime. The incorrect syntax will cause the code to fail.

What are Tokens?

- C program consists of various tokens.

Ex. - **Keyword, Identifier, Constant, Literal, Operators, or a Symbol.**



Identifiers

- Identifier is a name used to identify variable, function, or any user-defined item.

**Ex. - num1, getchar(), sum, ab_c
etc.**

```
num1 = 3;
```

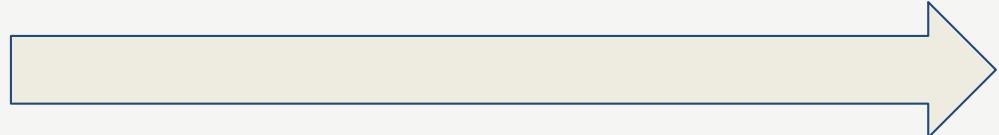
Identifiers

- Rules for writing the names of Identifiers
 - An identifier starts with a letter A to Z, a to z,
 - or an underscore '_'.
Followed by zero or more letters, underscores, and digits (0 to 9).

```
num1 = 3;
```

Keywords

- Some Reserved keywords names given in the next slide cannot be used as identifier names.



Keyword

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Strings

- Strings in C are always represented as a set of characters having null character at the end of the string.

Index	0	1	2	3	4	5
Variable	H	e	I	I	O	\0

Operators

- An operator is a symbol that tells the compiler to perform specific mathematical or logical functions.

Ex. - **+, -, *, /, ==, ++, --**

etc

a= b + c;

Special Characters

- Various Punctuators are used as a part of Syntax

Ex. - [], (), { }, #, *
etc.

return 0;

Constants

- A constant is a value assigned to variable which will remain the throughout the program

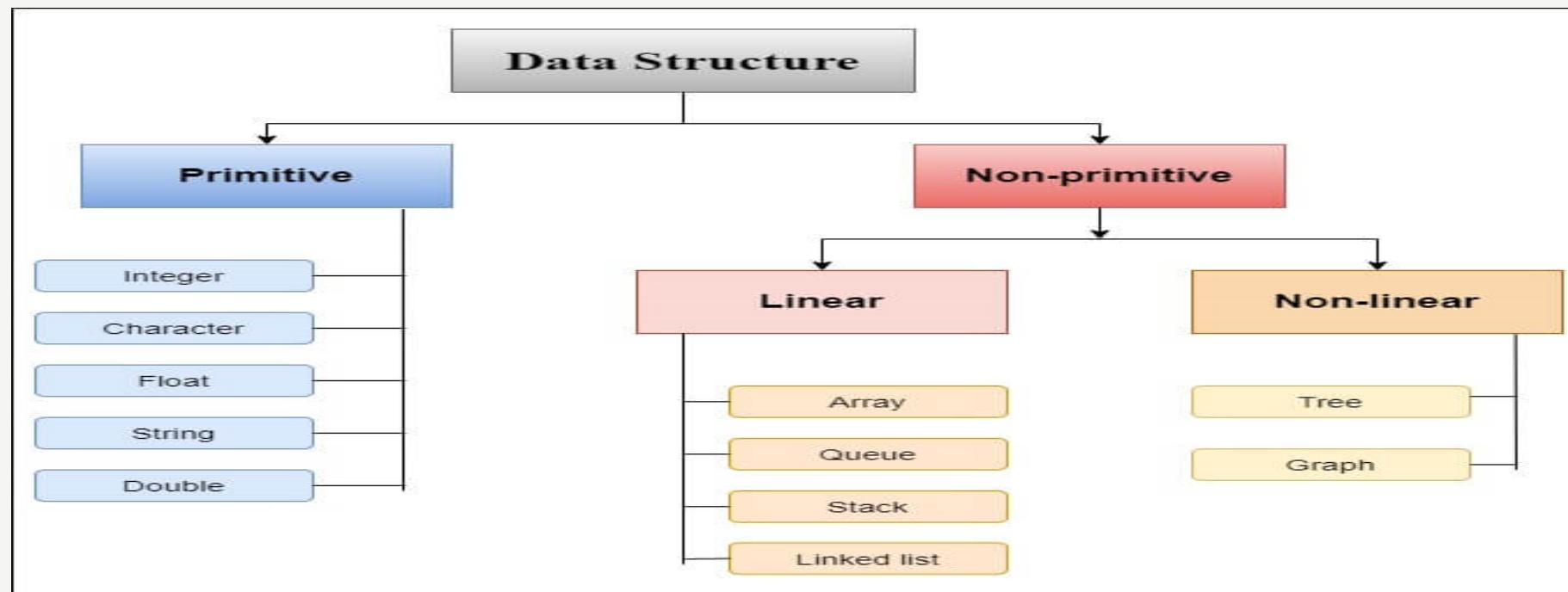
```
const int abc = 5;
```

2.Data Structures

Data Structures

2. Data structures

- A **data structure** is a particular way of organizing data in a computer so that it can be used effectively.



What are Data Types?

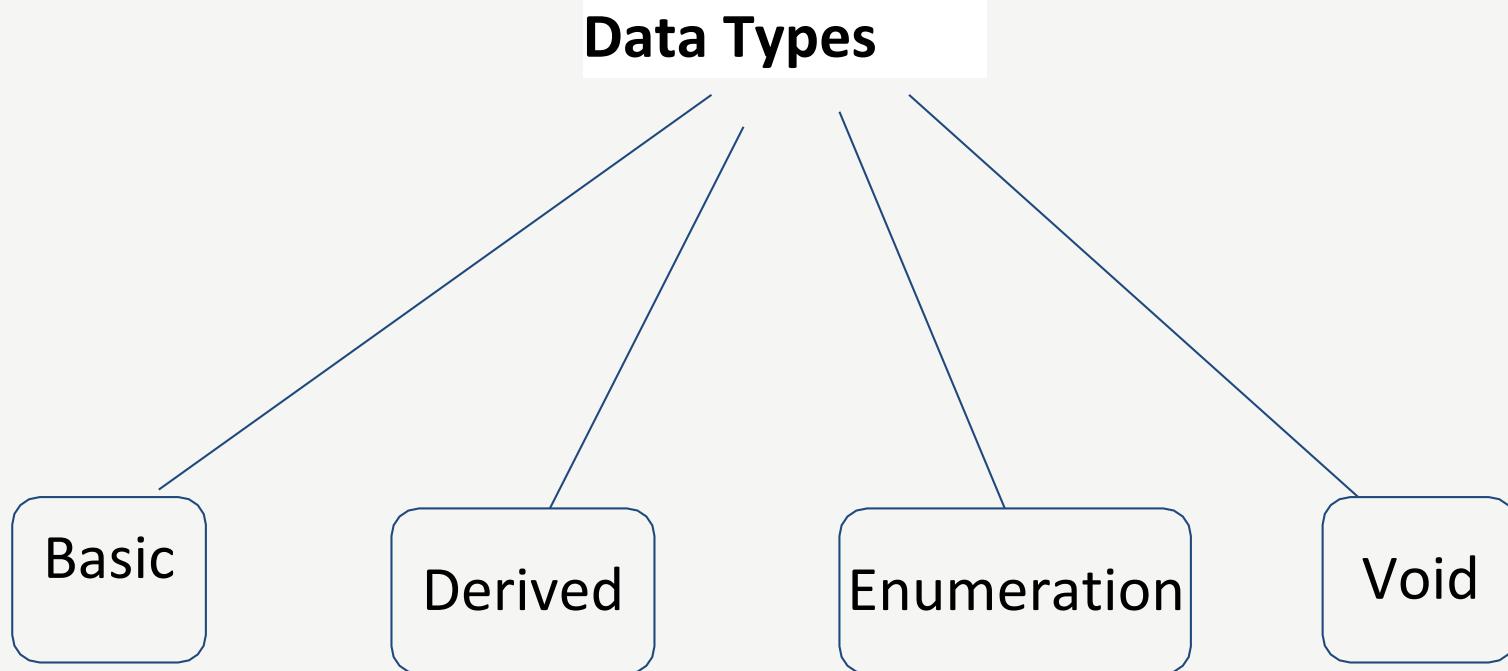
- A data type specifies what type of data a variable can store such as integer, floating, character, etc.

Ex. - int, float, char

etc.

```
int res =  
5;
```

Types of Data Types



Basic Data Types

- The basic data types are integer- and floating-point based.

```
float zab1 = 5.34;
```

```
int res = 5;
```

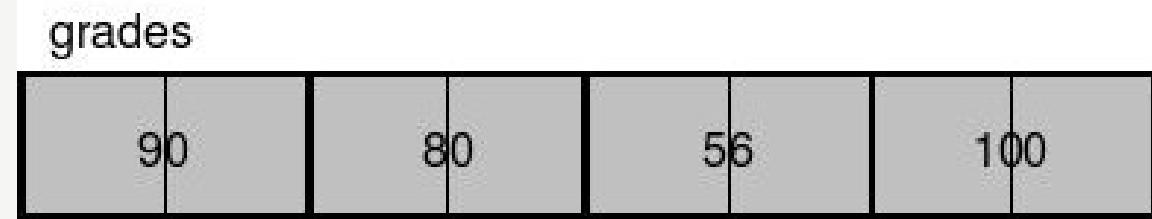
```
char b = 'G';
```

Basic Data Types

Data Types	Memory Size	Range
short int	2 byte	-32,768 to 32,767
signed short int	2 byte	-32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	
double	8 byte	
long double	10 byte	

Derived Data Types

- Array
 - S
 - Pointers
 - Union
- Structure



```
int * p = &n;
```

Void

- Void is an empty data type that has no value.

```
void num();
```

Refer This Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/C/dataType.c>

<https://github.com/TopsCode/Software-Engineering/blob/master/C/inputOutput1.c>

<https://github.com/TopsCode/Software-Engineering/blob/master/C/simpleProgram.c>

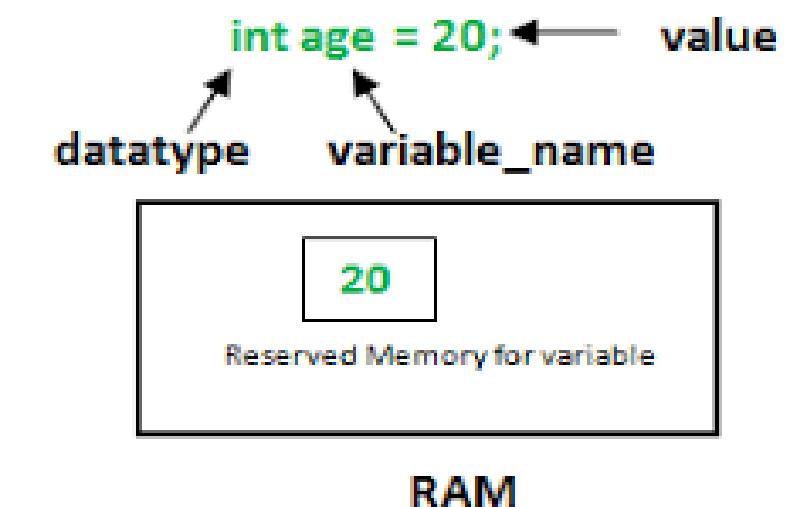
<https://github.com/TopsCode/Software-Engineering/blob/master/C/inputOutput.c>

3. Variables

Variables

3. Variables

- Variables are the names you give to computer memory locations which are used to store values in a computer program.
- For example, assume you want to store two values 10 and 20 in your program and at a later stage, you want to use these two values. Let's see how you will do it. Here are the following three simple steps –
 - i. Create variables with appropriate names.
 - ii. Store your values in those two variables.
 - iii. Retrieve and use the stored values from the variables.

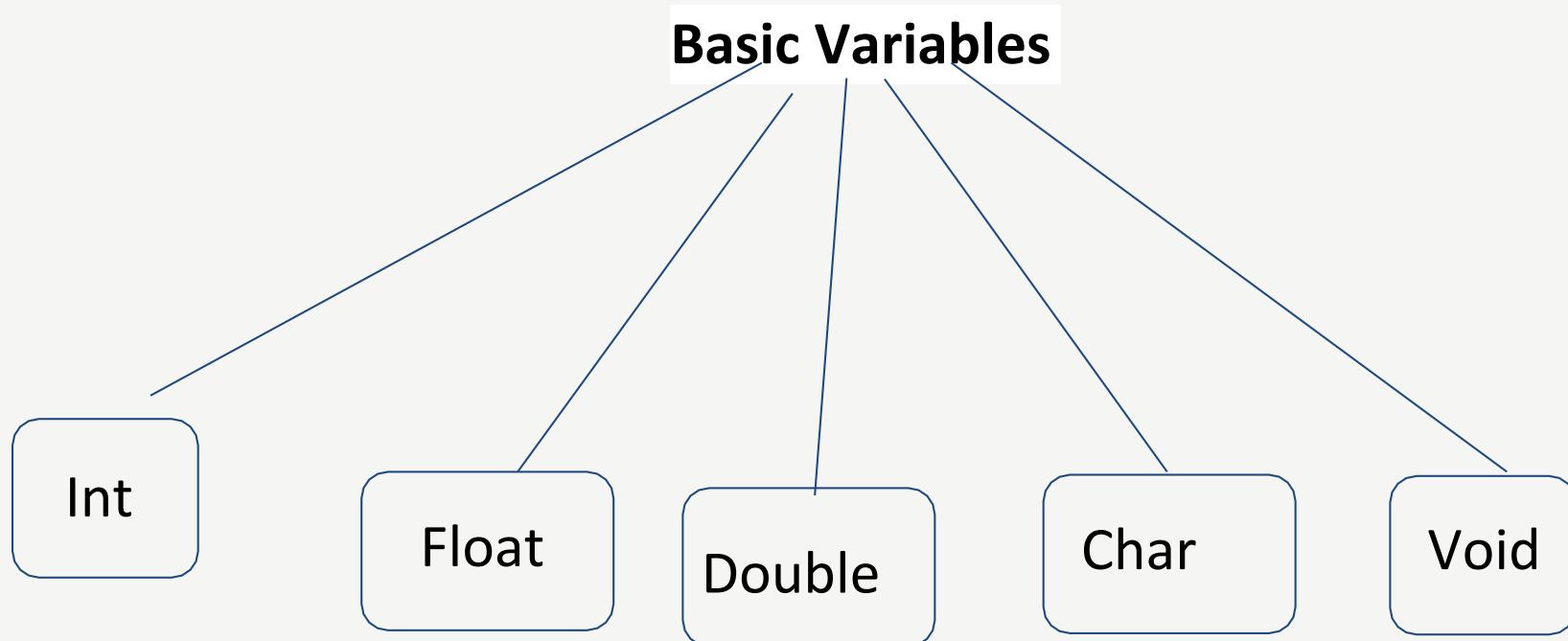


Deep Dive in Variables

- A variable is a name given to a storage area our programs can use to store values.
- The name of a variable can be composed letters, digits, and the underscore character.

```
int res =  
5;
```

Basic Types of Variables



Variable Declaration

- A variable declaration tells the where and how much storage to for the variable.

```
float f,  
salary;  
double d;
```

Variable Definition

- A variable definition assigns a value in variable
 - .

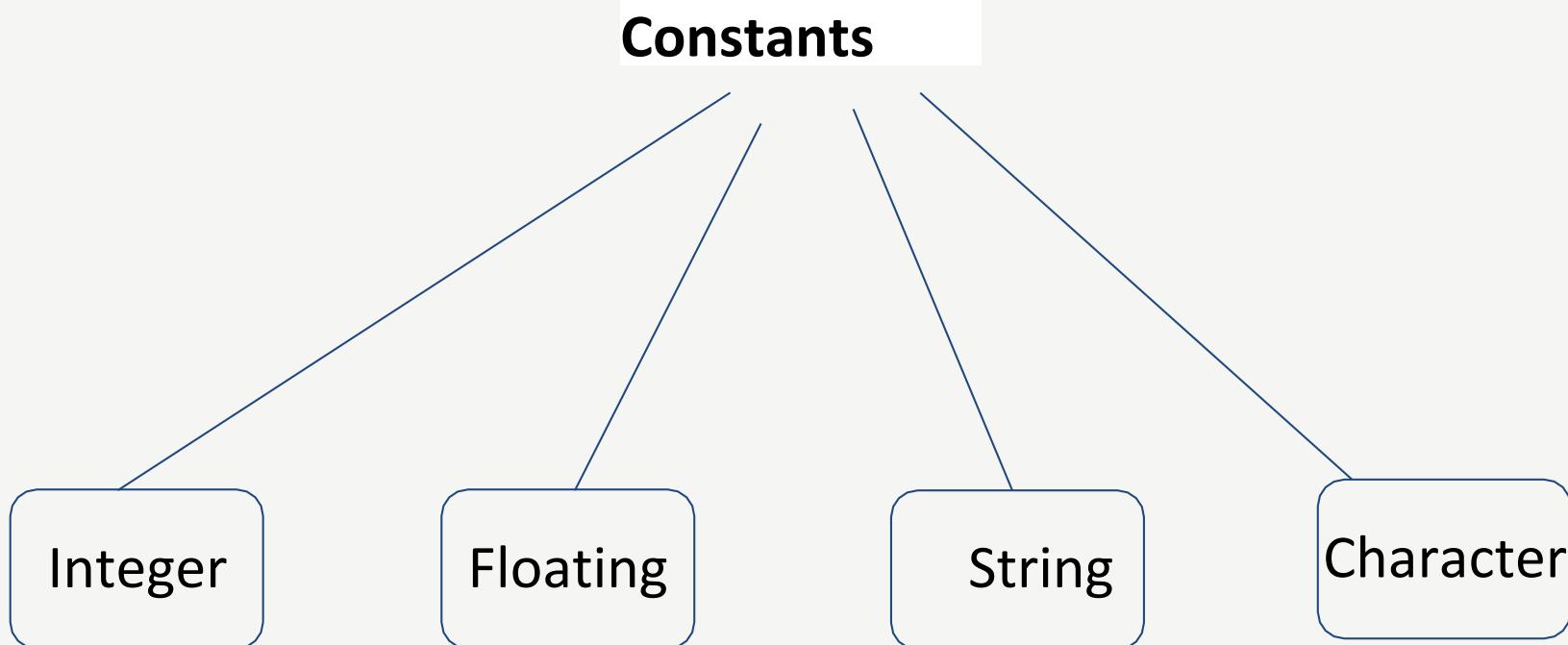
i = 43;

What are Constants?

- Constants are the fixed value in program . Which means that we cannot change it's value.

```
const int res = 5;
```

Types of Constants



const keyword

- Using const keyword
- Syntax : const data_type variable_name = value ;

Ex. - const int a =

10;

Refer This Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2B/variable/variables.cpp>

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2B/variable/globalVariable.cpp>

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2B/variable/localVariable.cpp>

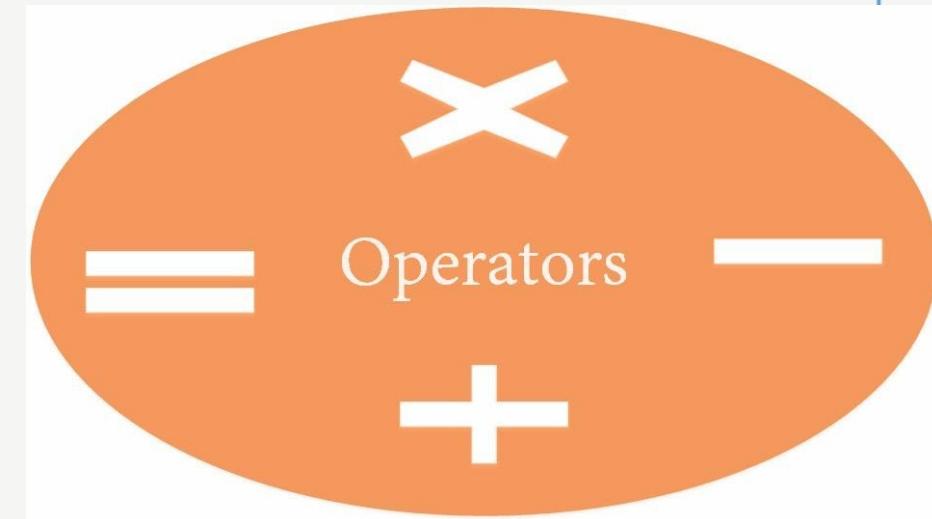
<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2B/variable/typeCasting.cpp>

4. Operators

What are Operators?

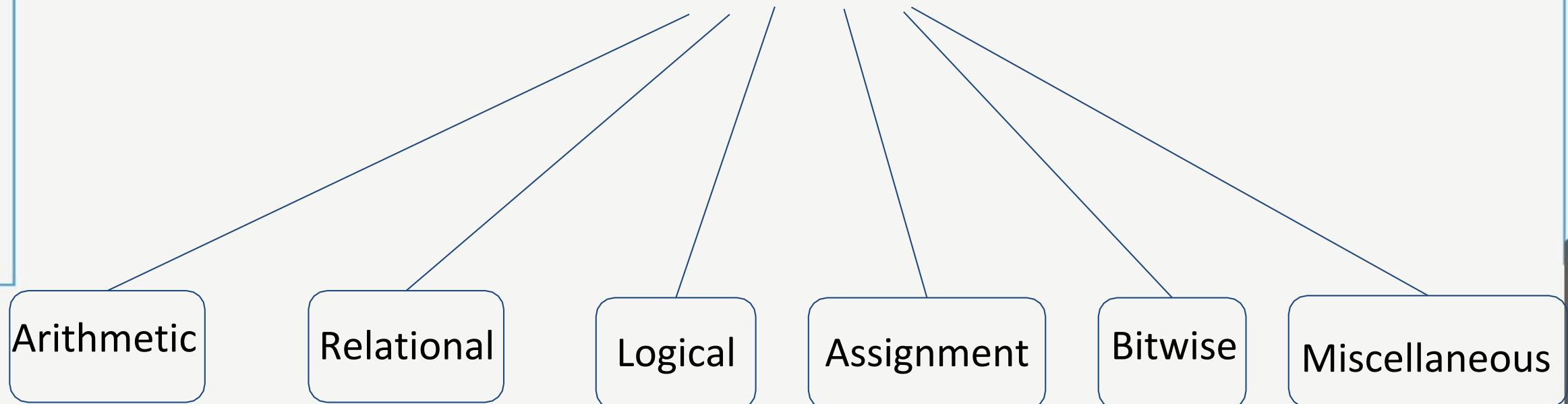
- A symbol that takes one or more operands such as variables, expressions or values and operates on them to give an output.

Ex. - =, +, -, /, *, ==, ++, --, %,
etc.



Types of Operators

Operators



Arithmetic Operators

Operator	Function	Example
+	Addition	<code>var=a+b</code>
-	Subtraction	<code>var=a-b</code>
*	Multiplication	<code>var=a*b</code>
/	Division	<code>var=a/b</code>
%	Modulo	<code>var=a%b</code>
++	Increment	<code>var++</code>
--	Decrement	<code>var-</code>

Relational Operators

Operator	Meaning of Operator	Example
<code>==</code>	Equal to	$5 == 3$ is evaluated to 0
<code>></code>	Greater than	$5 > 3$ is evaluated to 1
<code><</code>	Less than	$5 < 3$ is evaluated to 0
<code>!=</code>	Not equal to	$5 != 3$ is evaluated to 1
<code>>=</code>	Greater than or equal to	$5 >= 3$ is evaluated to 1
<code><=</code>	Less than or equal to	$5 <= 3$ is evaluated to 0

Logical Operators

Operator	Meaning	Example
<code>&&</code>	Logical AND. True only if all operands are true	If $c = 5$ and $d = 2$ then, expression $((c==5) \&\& (d>5))$ equals to 0.
<code> </code>	Logical OR. True only if either one operand is true	If $c = 5$ and $d = 2$ then, expression $((c==5) \mid\mid (d>5))$ equals to 1.
<code>!</code>	Logical NOT. True only if the operand is 0	If $c = 5$ then, expression $!(c==5)$ equals to 0.

Assignment Operators

- The Basic type of Assignment operator is '='.
- There are other derived operators

Ex. - *=, -=, /=, += etc.

C += 15;

C = C + 15;

Bitwise Operators

Operators	Meaning of operators
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

Miscellaneous Operators

Operator	Description	Example
sizeof()	Returns the size of a variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of a variable.	&a; returns the actual address of the variable.
*	Pointer to a variable.	*a;
?:	Conditional Expression.	If Condition is true ? then value X : otherwise value Y

Refer this Example:

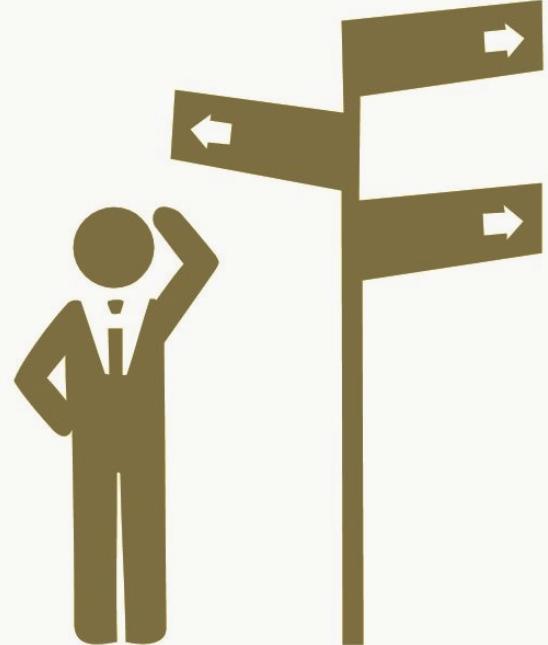
- **Assignment Operator**
<https://github.com/TopsCode/Software-Engineering/blob/master/C/AssignmentOperator.c>
- **Arithmetic Operator Example**
<https://github.com/TopsCode/Software-Engineering/blob/master/C/ArthmeticOperator.c>
- **Logical Operator Example**
<https://github.com/TopsCode/Software-Engineering/blob/master/C/logicalOperator.c>

5. Control & Looping Structures

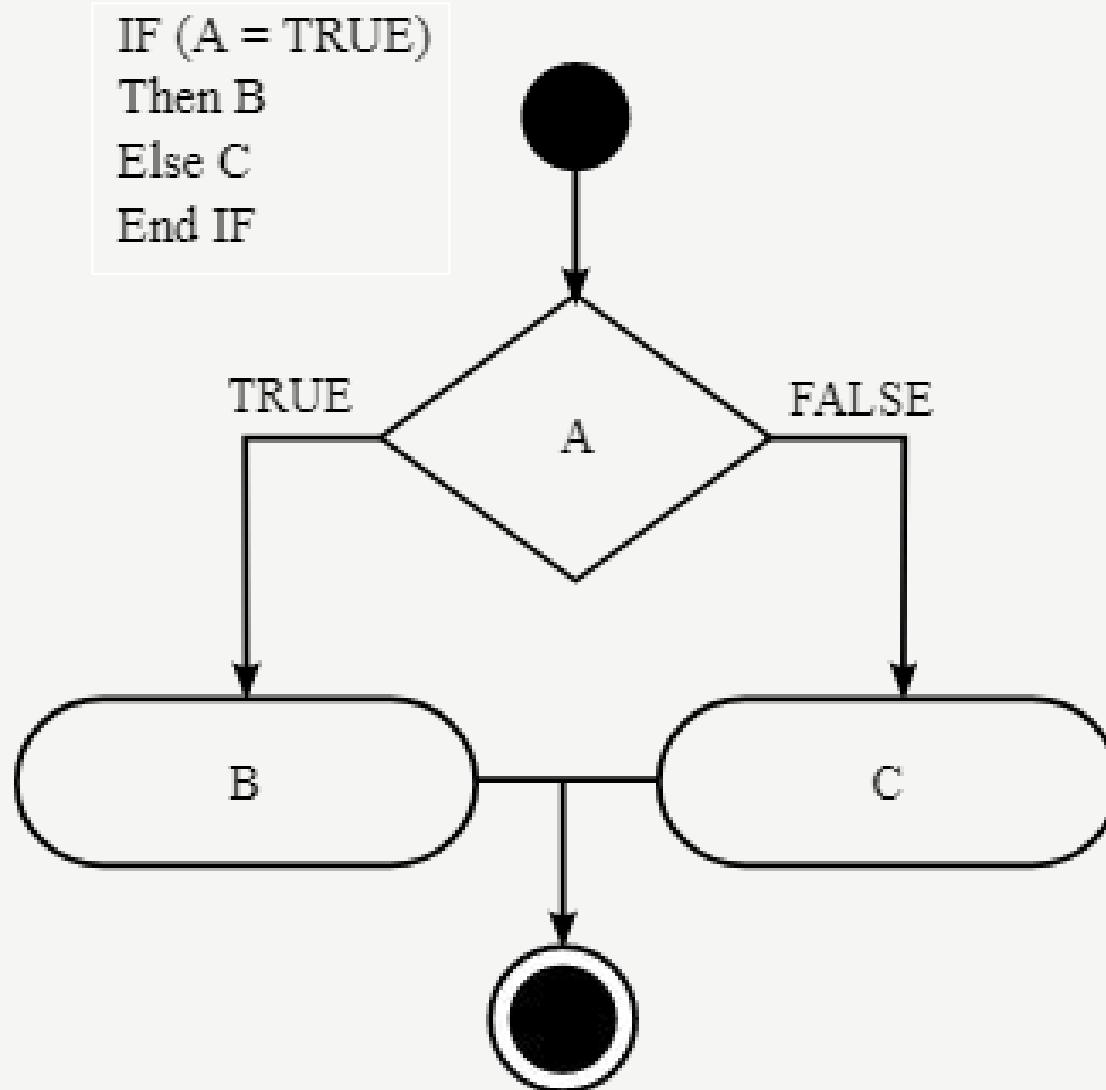
What is Control Structure?

- Real life situations where we have to condition based decisions by asking 'if' questions.

**Ex. - if age is above 18, I am allowed
drive vehicles or else not
allowed.**



Flow of Control Structure



Control structure Statements

- Following types of Statements:
 - if
 - if else
 - Nested if
 - Switch
 - Nested switch

If Statements

- if statement is the basic decision making statement
- Used to decide whether a certain statement or block of statements will be executed or not

If Statements

- Syntax :

```
if( condition )  
{  
    statement_1 ; // true block  
    statements  
}  
statement x ;
```

Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/C/if.c>

If else Statements

- if else statement allows selecting any one of the two available options depending upon the output of the test condition

If else Statements

- **Syntax :**

```
if (condition )  
{  
    statements; // true  
}           statement  
else  
{  
    statements ; // false statement  
}
```

Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/C/ifElse.c>

Nested If Statements

- Nested if statement is simply an if statement embedded with another if statement

Nested If Statements

- Syntax

:

```
if (condition1)
{
    statements ;      // executes when condition1 is
    if ( condition2) true
    {
        statements ;      // executes when condition2 is
        true
    }
}
```

Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/C/ifElseLadder.c>

Refer this Example:

- **If statement:**

<https://github.com/TopsCode/Software-Engineering/blob/master/C/if.c>

- **If-else:**

<https://github.com/TopsCode/Software-Engineering/blob/master/C/ifElse.c>

- **If else with compound Relational Test:**

<https://github.com/TopsCode/Software-Engineering/blob/master/C/compoundRelationalTest.c>

- **Nested if-else**

<https://github.com/TopsCode/Software-Engineering/blob/master/C/NestedIfElse.c>

- **If-else Ladder:**

<https://github.com/TopsCode/Software-Engineering/blob/master/C/ifElseLadder.c>

Switch Statements

- Switch case statements are a substitute for long if statements that compare a variable to several integer values

Switch Statements

- **Syntax**

:

```
switch ( n )          // executed when n =  
{  
    case 1 : break;   // executed when n =  
    case 2 : break ;  
    default :         // executed when n doesn't match any  
                      case  
}
```

Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/C/switchCase.c>

Nested Switch Statements

- Nested Switch Statements occurs when a switch statement is defined inside another switch statement.

Nested Switch Statements

- Syntax :

```
switch(ch1)
{
    case 'A': printf ("\n This A is part of outer switch " );
    switch (ch2)
    {
        case 'A ': printf("\n This A is part of inner switch ");
        break;
        case' B' :
            }
        break ;
        case' B' :
    }
```

Looping Structures

What are Loops?

- A loop statement allows us to execute a statement or group of statements multiple times based on a condition

**Ex. - printing 1 to 100 on the
output screen**



Types of Loops

- **Entry Controlled** A condition is checked before executing the loop. It is also called as a pre-checking loop.
- **Exit Controlled** A condition is checked after executing the loop. It is called as a post-checking loop.

Entry controlled Loops

1. **For Loops-** It is a repetition control structure that allows you to efficiently write loop that needs to execute a specific number of times.

2. **While Loops-** It repeatedly executes a target statement as long as the condition is true.

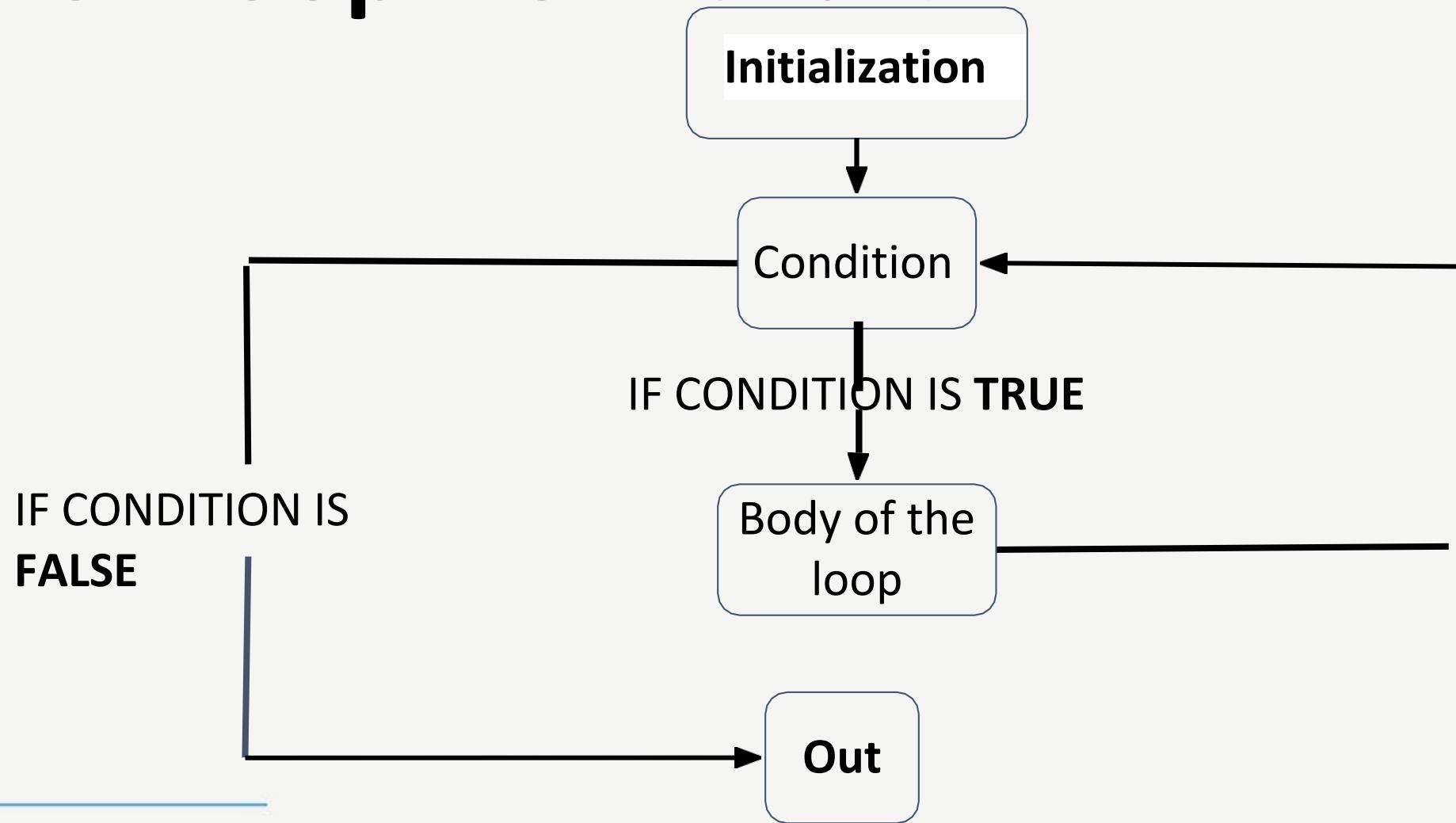
For Loops

- Used to efficiently write a loop that needs to execute a specific number of times.

Syntax :

```
for ( initialization ; test condition ; increment )  
{  
    Body of loop  
}
```

For Loop Flow Chart



For Loop

- All for loops are executed in following sequence
 - : Step 1 : It executes initialization statements
 - Step 2 : It checks the condition;
 - if true then go to Step 3
 - otherwise Step 4
 - Step 3: Executes loop and go to Step 2
 - Step 4 : Out of the Loop

Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/C/ForLoop.c>

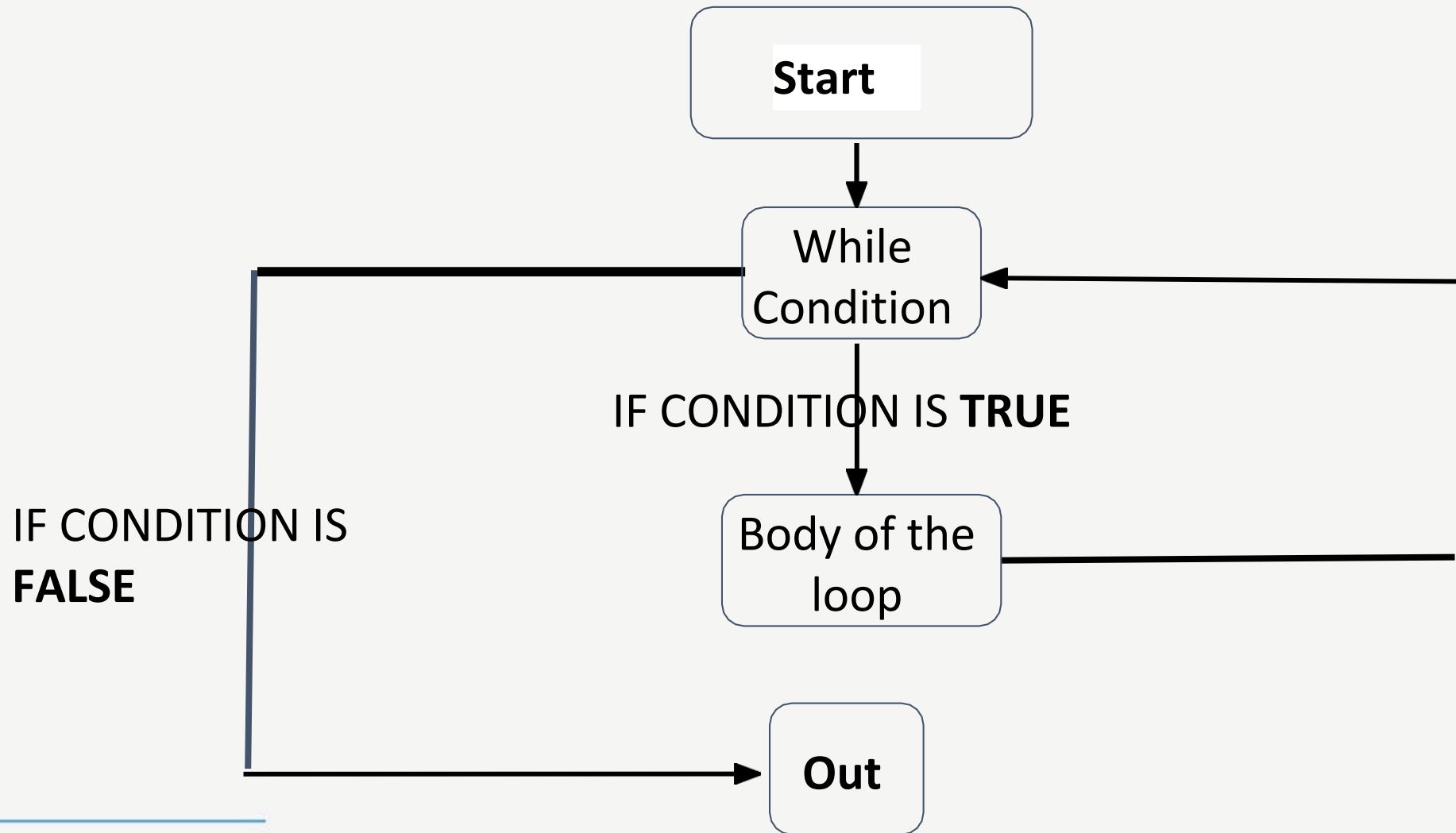
While Loops

- It repeatedly executes a target statement as long as the given condition is true

Syntax :

```
while (condition )  
{  
    Body of loop  
}
```

While Loop Flow Chart



While Loops

- All while loops are executed in following sequence

:

Step 1 : It checks the test

condition if true then go to

Step 2 otherwise to Step 3

Step 2 : Executes body of loop and go to Step

1. Step 3 : Other statements of program.

Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/C/while.c>

Exit controlled Loops

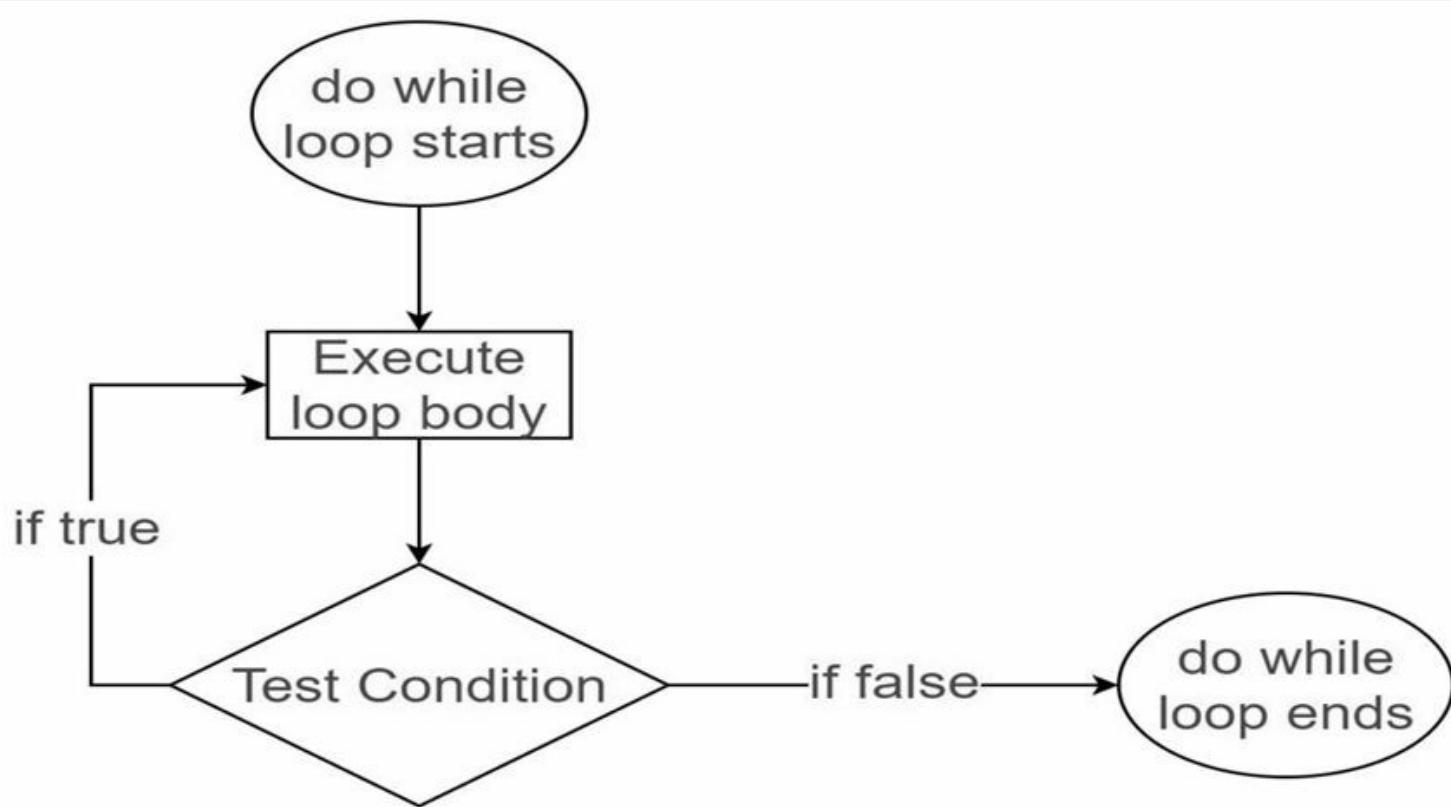
1. **Do-while Loops-** Do-while loop is similar to while loop , except the fact that it execute once even ~~if~~ condition is false.

Do-while Loops

- **Syntax :**

```
do
{
    body of loop
} while (condition);
```

Do-while Loops



Do-while Loops

- All do while loops are executed in following sequence :
 - Step 1 : Executes the body of loop and go to Step 2
 - .
 - Step 2 : It checks the test condition if true then go to Step 1 otherwise go to Step 3.
 - Step 3: Other statements of the program .

Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/C/DoWhile.c>

Loops

Jumping Statements:

1. Goto Statement.
2. Break Statement.
3. Continue Statement

Loops

The GOTO statement:

- By using this goto statements we can transfer the control from current location to anywhere in the program.
- To do all this we have to specify a label with goto and the control will transfer to the location where the label is specified.

Refer this Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/C/got.c>

goto label;

label:

Loops

The Break Statement:

- The break statement is used inside loop or switch statement.
- When compiler finds the break statement inside a loop, compiler will abort the loop and continue to execute statements followed by loop.

Syntax: break ;

Refer this Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/C/Break.c>

Loops

Continue statement:

- The continue statement is also used inside loop.
- When compiler finds the continue statement inside a loop, compiler will skip all the following statements in the loop and resume the next loop iteration.

Syntax: continue ;

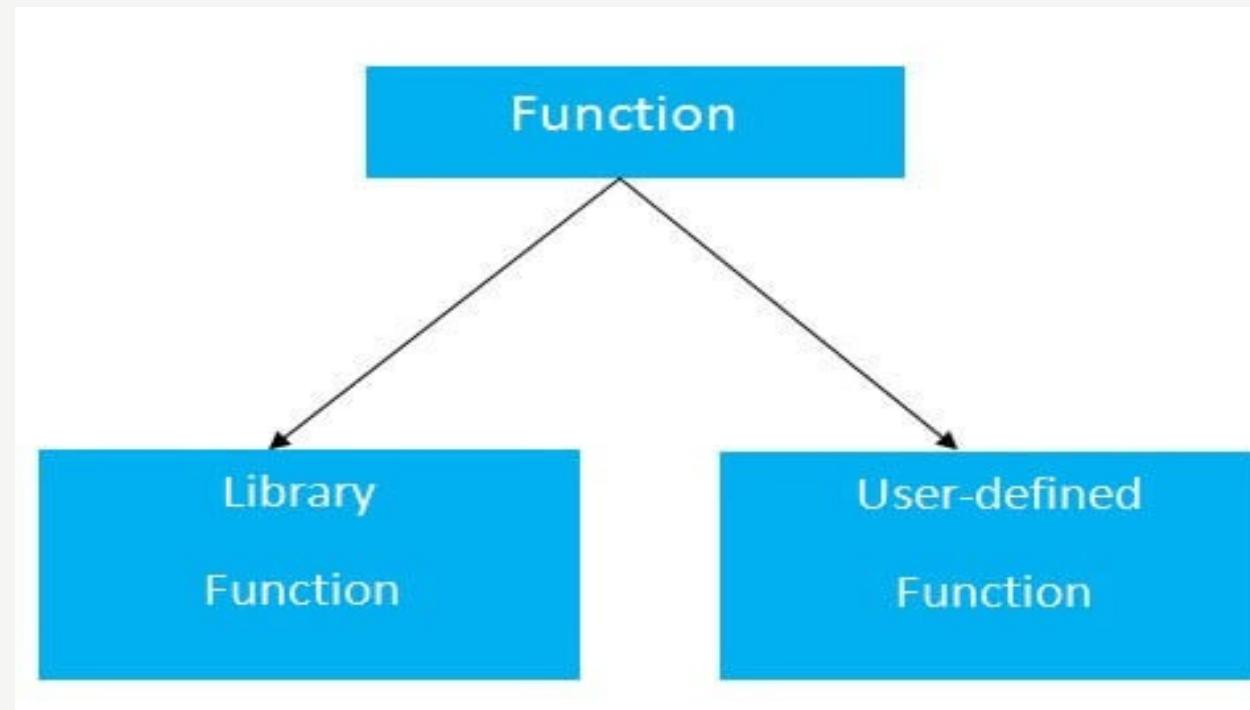
Refer this Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/C/continue.c>

6. Functions

6. Functions

- A function is a block of organized code that is used to perform a single task. They provide better modularity for your application and reuse-ability. Depending on the programming language, a function may be called a subroutine, a procedure, a routine, a method, or a subprogram.



Types of Parameters

**Parameter in
Functions**

Call by
Value

Call by
Reference

Call by Value

- A method of passing parameters, where it copies the actual value into formal parameter
- Changes made to the parameter inside the function have no effect on the actual parameters

Call by Reference

- A method of passing arguments which copies the address of an argument into formal parameter.
- Changes made to the parameter affect the passed argument.

What are Functions?

- A function is a set of statements that take inputs, do some specific computation and produces output

Ex. - main(), sum(), swap()

etc.

```
int num1(  
);
```

Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/C/Function1.c>

Parts of Functions

- **FUNCTIONS-**
 - Function Definition
 - Function Call
 - Function Declaration

```
int num1(  
);
```

Functions Definition

- Syntax of Function Definition-

```
return_typefunction_name( parameter list )  
{  
    body of the function  
}
```

Components in Definition

Return_type : It is data type of value which function will return. If function does not return any value data type will be void().

```
return_type  function_name( parameter list )  
{  
    body of the function  
}
```

Components in Definition

Function_name : It is the name given to the function by the programmer.

```
return_type function_name( parameter list )  
{  
    body of the function  
}
```

Components in Definition

Parameter list : This list refers to type, order and number of parameters of the function. A function can have no parameters also.

```
return_type function_name( parameter list  
)  
{  
body of the function  
}
```

Components in Definition

Body of function : It is collection of statements that define the working of the function.

```
return_type function_name( parameter list )  
{  
    body of the function  
}
```

Function Declaration

- It tells the compiler about the function name and how to call the function.
- Syntax :
`return_type function_name (parameter_list) ;`

Function Declaration

- Only type is required in function declaration , we can skip the parameter name
- Example:

```
int add ( int , int );
```

Function Call

- To use a function, we have to call that function to perform the given task.

Function Call

- When a program calls a function, the compiler gets redirected towards the function definition
- Function Call simply pass the required parameters along with the function name

What are Function Parameters

- Parameters are the variables that are taken as input to perform the function

```
int max(int num1, int  
        num2);
```

Scope of a Variable

What is Scope?

- Scope is the part of the program where a defined variable can have its existence and beyond that it cannot exist

Three Scopes

- **Three places where variables can be declared-**
 - Local Variables : Declared inside a block
 - Global Variables : Declared outside all functions
 - Formal Parameter : Declared in function definition

Local Variables

```
#include<stdio.h>
> void
{   main(){ x , y , z           //      local
                ;             variable
                x=10 ;
                y =20 ;
                z =x + y;    %d      %d ” , x, y,
}   printf(“ %d  z );
```

Global Variables

```
#include<stdio.h>
> int z;                                // global variable
void main ()
{
    int x , y ;                         // local variable
    x=10 ;
    y =20 ;
    z =x + y ;
    printf(“ %d  %d  %d ” , x, y, z );
}
```

Formal Parameters

```
void swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

Refer This Example

[https://github.com/TopsCode/Software-
Engineering/blob/master/C/functions/pass_by_reference.c](https://github.com/TopsCode/Software-Engineering/blob/master/C/functions/pass_by_reference.c)

[https://github.com/TopsCode/Software-
Engineering/blob/master/C/functions/pass_by_value.c](https://github.com/TopsCode/Software-Engineering/blob/master/C/functions/pass_by_value.c)

7. Arrays & Strings

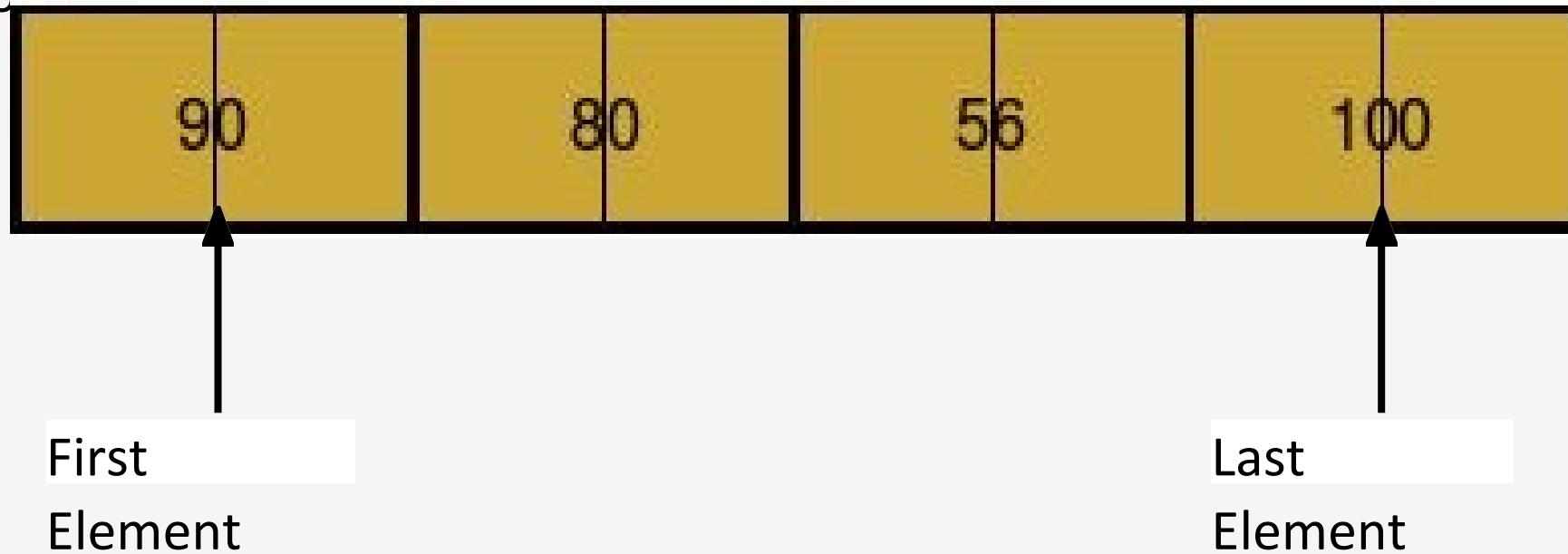
What are Arrays?

- An array is used to store a collection of data, and it is often used as a collection variables of the same type.



What are Arrays?

- All arrays consist of contiguous memory locations



Declaring Arrays

- In declaration we specify the type of element and size of the array element
- **Syntax :**

Ex. - data type array_name [size] ;
 int roll[20
];

Initializing Arrays

- We can initialize an array in C either one by one or using single statement

Ex. - **double balance [] = { 1000.0, 2.0, 3.4, 7.0, 50.0};**

OR

balance[4] = 50.0 ;

Accessing Arrays

Accessing Array Elements

- An element is accessed by placing the index of the element within the square brackets after the name of the array

Ex. - **double income = balance[9];**

Accessing Struct Members

- The individual members of structure can be accessed using the member access operator (.)
- It connects the structure variable and the structure member.
- **Syntax:**

structure_variable.structure_member

<https://github.com/TopsCode/SoftwareEngineering/blob/master/c%2B%2B/Array/1dArray.cpp>

Types of Arrays

Types of Arrays

- Single/ One Dimensional Array
- Multi Dimensional Array

Single Dimensional

```
int roll[20];
```

Example :

<https://github.com/TopsCode/Software-Engineering/blob/master/C/Array/example1.c>

Multi Dimensional

```
int roll [20][12]...;
```

Multi Dimensional

- C programming language provides us multi-dimensional array.
- Out of which we will discuss Two Dimensional Array

Multidimensional Arrays

What are they?

- Multidimensional arrays are arrays of arrays
- **General Form-**
`data_type array_name[size1][size2]....[sizeN];`

Example

```
int x[2][3][4] =  
{  
    { {0,1,2,3}, {4,5,6,7}, {8,9,10,11} },  
    { {12,13,14,15}, {16,17,18,19}, {20,21,22,23} }  
};
```

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2B/Array/2DArray.cpp>

Strings

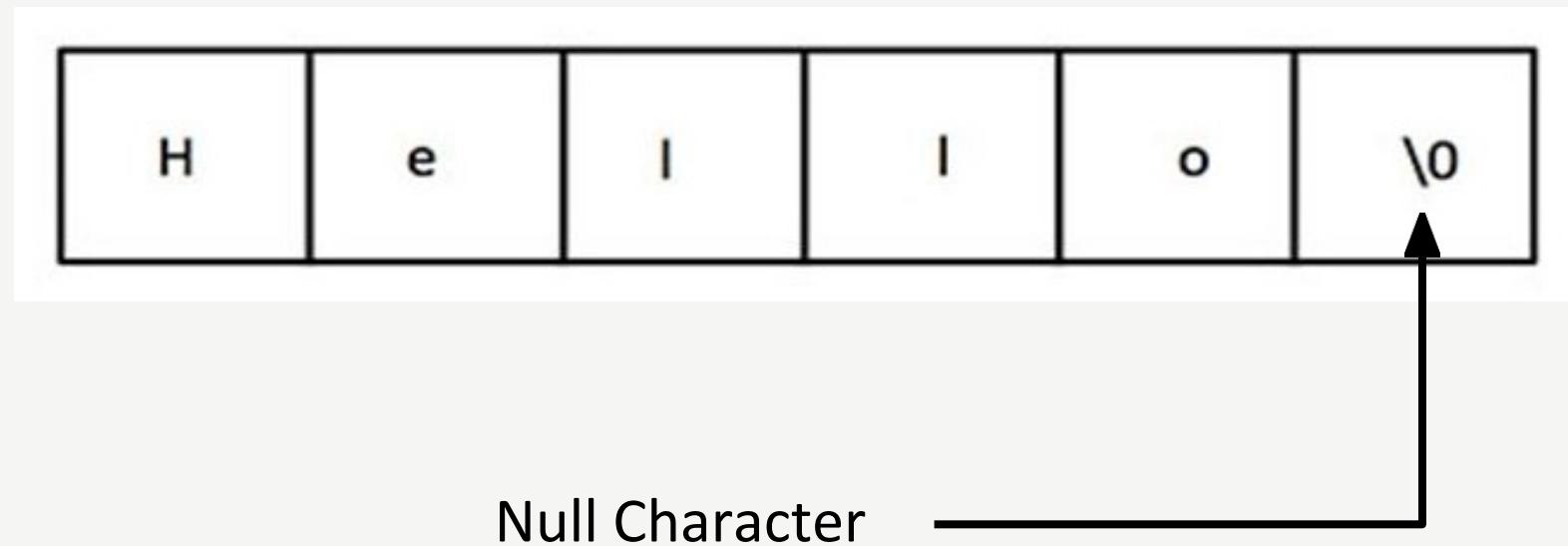
What are Strings?

- A string is an array of characters stored in a consecutive memory locations
- The ending character is always the null character '\0'. It acts as string terminator.
- **Syntax :**

```
char string_name [ length ] ;
```

Strings

- The compiler automatically places the '\0' at the end of the string when we initializes the array



String Functions

Function	What It Does
strcpy()	Copies one string to another.
strlen()	Returns the length of a string, not counting the NULL character

String Functions

Function	What It Does
strcmp()	Compares two strings. If the strings match, the function returns 0.
strcat()	Appends one string to another, creating a single string out of two.

String Functions

Refer this Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/C/strlen.c>

<https://github.com/TopsCode/Software-Engineering/blob/master/C/stringFunctin.c>

Module 3

[OOP Concepts]

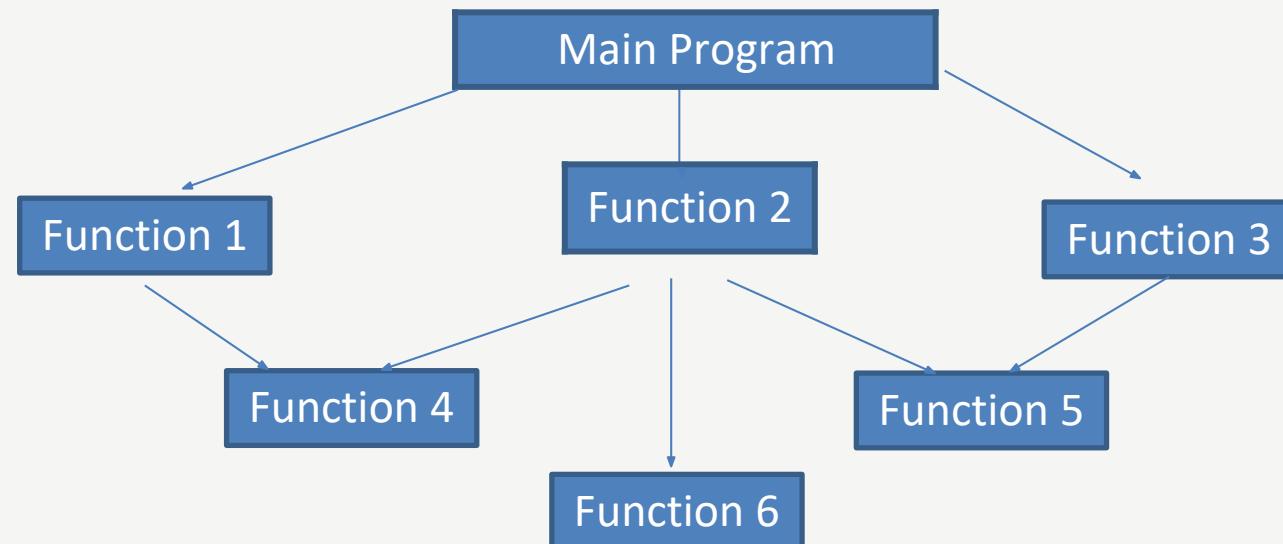
Procedure Oriented Programming & Object Oriented Programming

Procedure Oriented Programming

In Procedure Oriented programming , the problem is viewed as sequence of things to be done such as reading, calculating and printing.

The number of functions are return to accomplish such task, i.e. the focus is on functions.

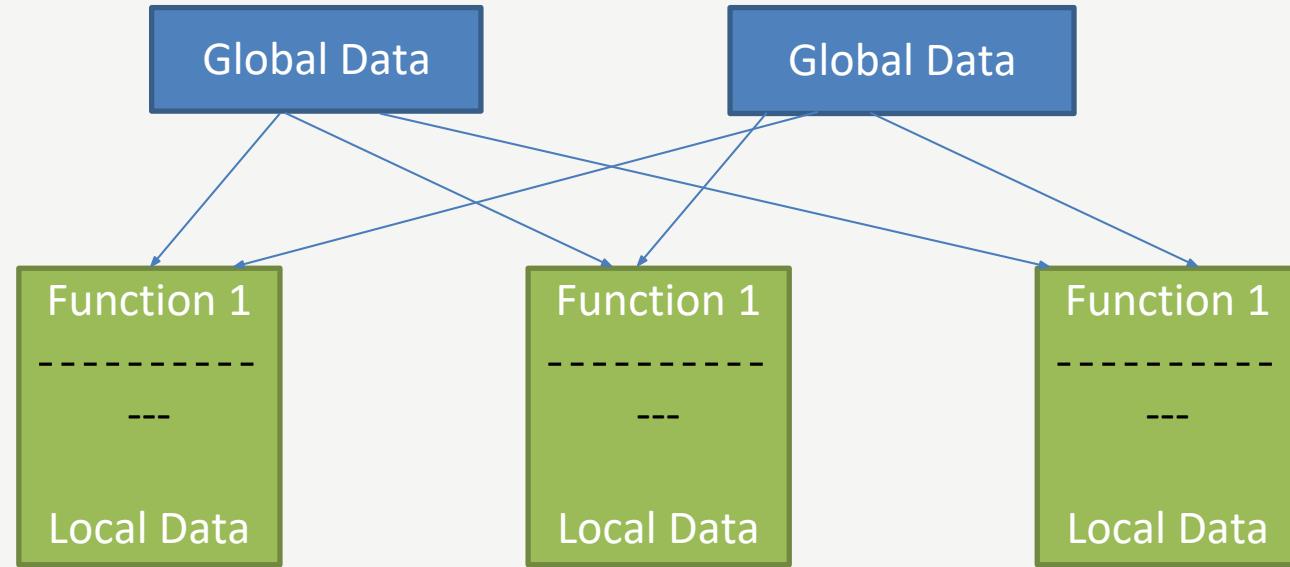
The typical program structure for procedure programming is shown below :



Procedure Oriented Programming

Characteristics of Procedure Oriented Programming.

- ❖ Emphasis is on doing things
- ❖ Large programs are broken into small known as functions
- ❖ Most of the function share global data.
- ❖ Data move openly around the system from function to function.
- ❖ Follows Top Down approach in program design



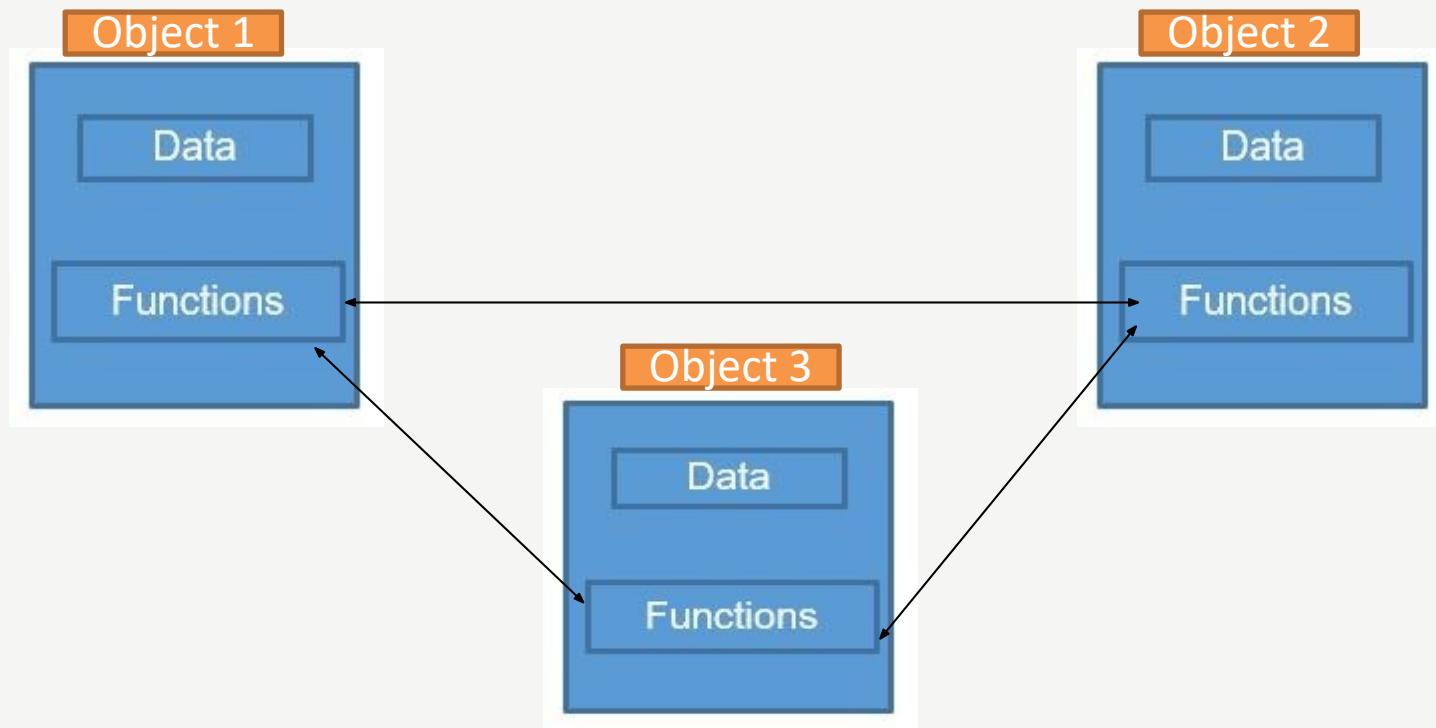
Object Oriented Programming

In order to remove some of the flaws of POP, OOP came into existence.

OOP treats data as critical element in program development and does not allow it to flow freely around the system.

It ties the data more closely to the function that operates on it.

Object Oriented Programming allows decomposition of program into a number of entities called objects and then builds data and function around these objects.



Object Oriented Programming

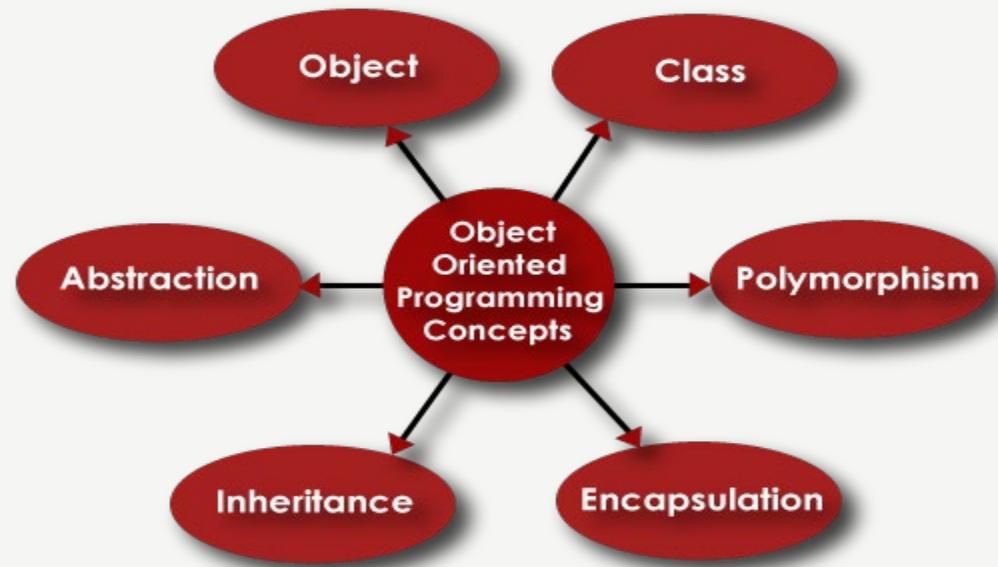
Characteristics of Object Oriented Programming:

- ❖ Emphasis on data rather than procedure.
- ❖ Program are divided into objects.
- ❖ Data is hidden and cannot be accessed by external functions.
- ❖ Objects may communicate with each other through functions.
- ❖ Follows bottom up approach in program design.

Basic Concepts of OOP

Some of the basic concepts of object oriented programming are:

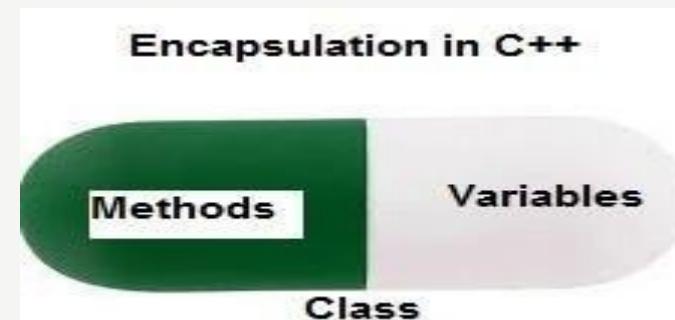
- ❖ Objects
- ❖ Classes
- ❖ Data abstraction and encapsulation
- ❖ Polymorphism
- ❖ Inheritance
- ❖ Dynamic Binding



Some of the basic concepts of object oriented programming are:

❖ **Encapsulation:**

In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.



❖ **Abstraction:**

Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

❖ Polymorphism:

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

C++ supports operator overloading and function overloading.

❖ Inheritance:

- The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.
- Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

❖ Classes

- These contain data and functions bundled together under a unit. In other words class is a collection of similar objects. When we define a class it just creates template or Skelton. So no memory is created when class is created. Memory is occupied only by object. for eg. :

Fruit is class of apple.

❖ Objects

- In other words object is an instance of a class.

NOTE: In other words classes acts as data types for objects.

❖ Member functions

- The functions defined inside the class as above are called member functions.

Example:

```
class classname
{
    variable declarations;
    Data Functions;
};

main ( )
{
    classname
    objectname1,objectname2,..
}
```

Types of accessifiers

Private	Public	Protected
Only for that class can access	Other class can also access.	Only immediate inheritance class can access.

	Direct-access scope	Class/object scope
Private	Declaring class	Declaring class
Protected	All derived classes	Declaring class
Friend	Derived in-project classes	Declaring project
Protected Friend	All derived classes	Declaring project
Public	All derived classes	All projects

Example of accessifiers:

```
Class classname
{
    private:
        datatype data;

    public:
        Member functions
};

main ()
{
    classname objectname1,objectname2,..;
}
```

- **How to Access Class Members**

It is possible to access the class members after a class is defined and objects are created.

General syntax to access class member:

```
Object_name.function_name (arguments);
```

```
class exforsys
{
    int a, b;
public:
    void sum(int,int);
} e1;
e1.sum(5,6);
```

• Scope Resolution Operator

Member functions can be defined within the class definition or separately using **scope resolution operator, ::**. Defining a member function within the class definition declares the function inline, even if you do not use the inline specifier. So either you can define Volume() function as below:

```
class Box
{
    public:

        double length; // Length of a box
        double breadth; // Breadth of a box
        double height; // Height of a box

        double getVolume(void)
        {
            return length * breadth * height;
        }
};
```

Refer This Example:

Simple Class:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/class/class1.cpp>

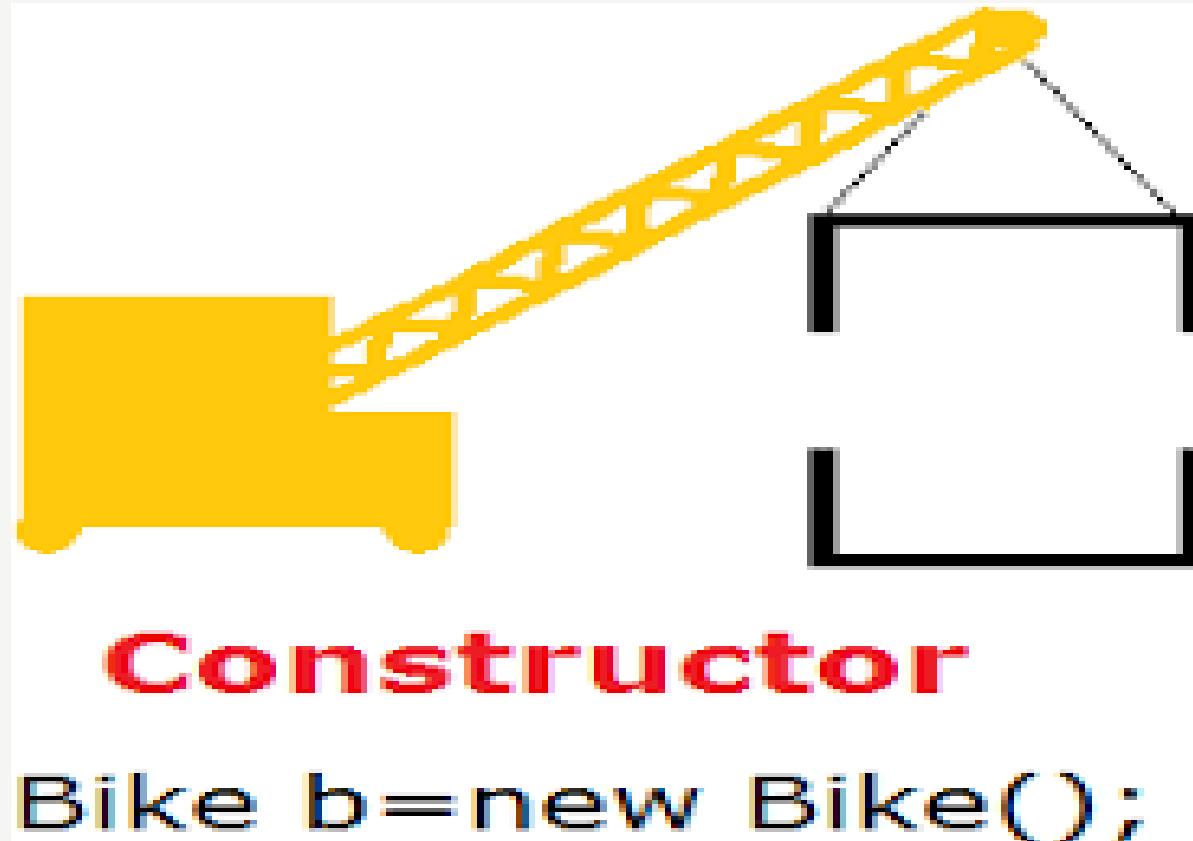
<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/class/class2.cpp>

Constructor

- It is a member function which initializes a class.
- A constructor has:
 - (i)the same name as the class itself
 - (ii)no return type
- A constructor is called **automatically** invoked whenever a new instance of a class is created.
- You must supply the arguments to the constructor when a new instance is created.
- If you do not specify a constructor, the compiler generates a default constructor for you (expects no parameters and has an empty body).

Types of Constructor

- Simple(Default) Constructor
- Parameterized Constructor
- Copy Constructor



Refer this Example:

Default Constructor:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Constructor/DefaultConstructor.cpp>

Parameterised Constructor:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Constructor/parameterisedConstructor.cpp>

Copy Constructor:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Constructor/copyConstructoar.cpp>

Dynamic Constructors

- The constructor can also be used to allocate memory while creating objects. This will enable the system to allocate the right amount for each object when the objects are not of the same size, thus resulting in the saving of memory.
- Allocation of memory to objects at the time of their construction is known as dynamic constructor of objects. The memory is allocated with the help of the new operator.

Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Constructor/DynamicConstructor.cpp>

What is a Destructor?

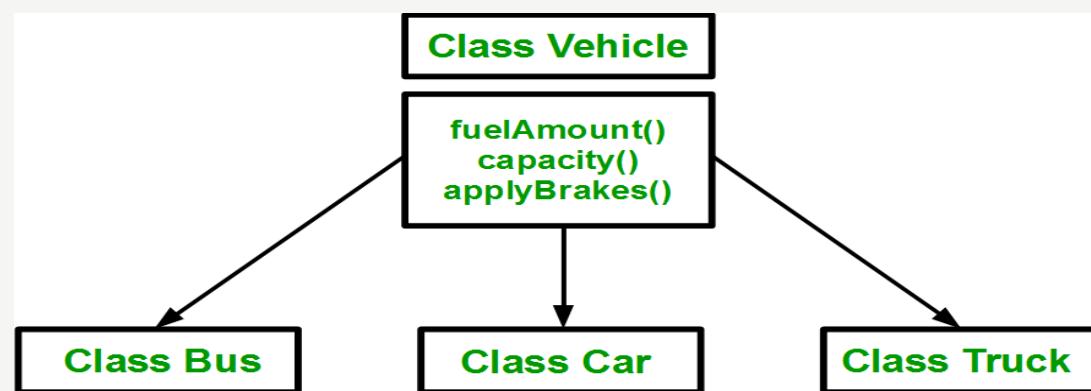
- It is a member function which deletes an object.
- A destructor function is called automatically when the object goes out of scope:
 1. the function ends
 1. the program ends
 2. a block containing temporary variables ends
 3. a delete operator is called

A destructor has:

1. the same name as the class but is preceded by a tilde (~)
2. no arguments and return no values

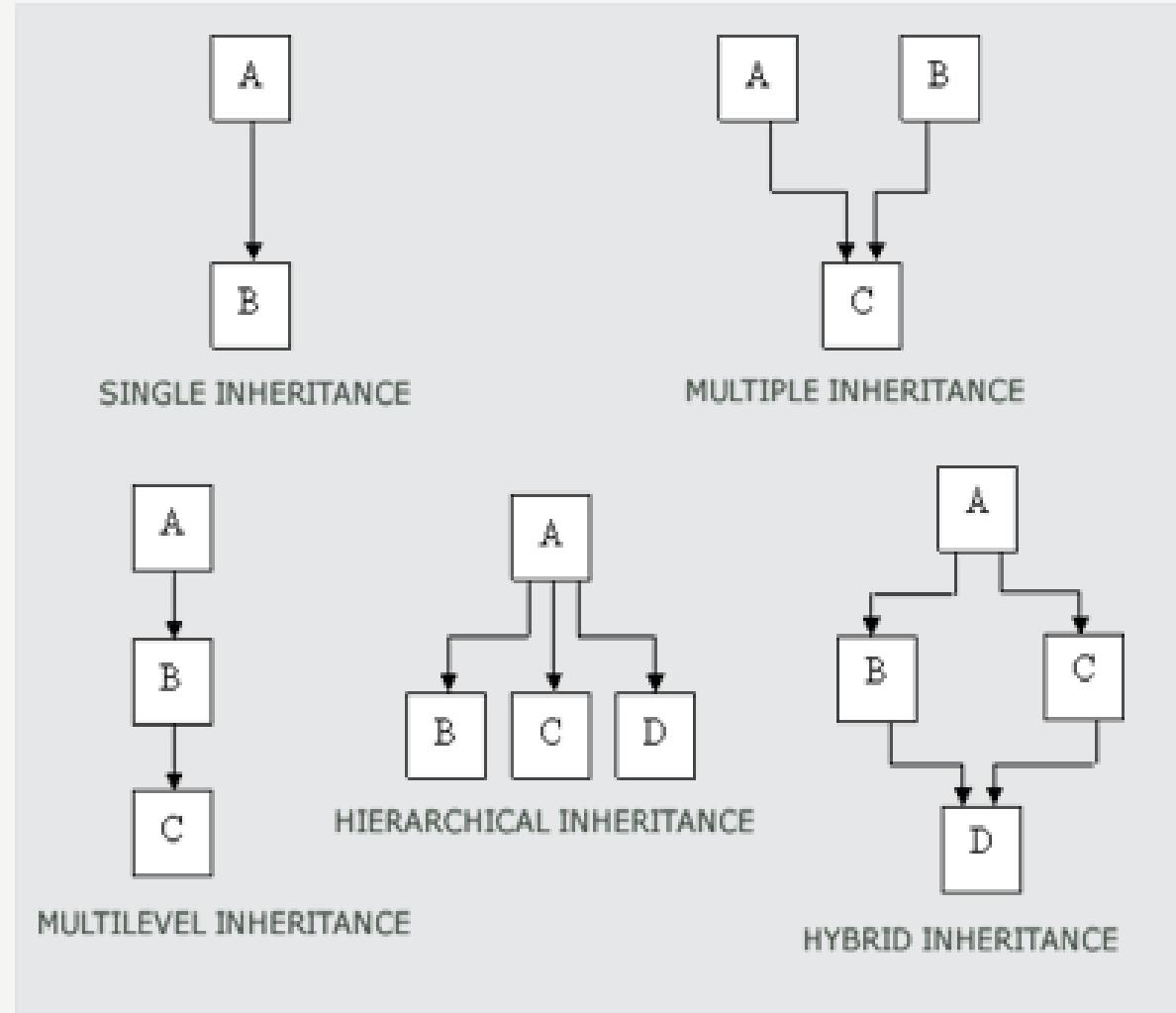
Inheritance

- Inheritance is a mechanism of reusing and extending existing classes without modifying them, thus producing hierarchical relationships between them.
- When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base class**, and the new class is referred to as the **derived class**.



Types of Inheritance

- Single Inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Multilevel Inheritance
- Hybrid Inheritance



Refer This Examples:

Single Inheritance:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Inheritance/single.cpp>

Multiple Inheritance:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Inheritance/multiple.cpp>

Multilevel Inheritance:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Inheritance/multilevel.cpp>

Hierarchical Inheritance

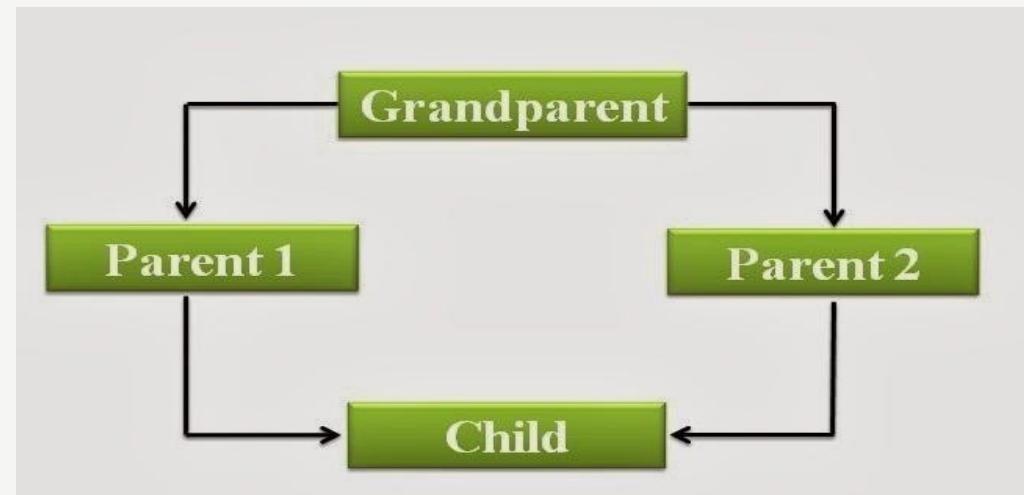
<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Inheritance/hierarchical.cpp>

Hybrid Inheritance

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Inheritance/hybrid.cpp>

Virtual Base Classes

- Virtual base class is used in situation where a derived have multiple copies of base class.
- When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class.



- Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/class/virtualClass.cpp>

Constructor in Derived Class

- Base class constructors are automatically called for you if they have no argument.
- If you want to call a superclass constructor with an argument, you must use the subclass's constructor initialization list.
- Unlike Java, **C++ supports multiple inheritance** (for better or worse), so the base class must be referred to by name, rather than "super()".

Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Constructor/constructorInheritance.cpp>

Polymorphism and Overloading

- Poly refers many.
- "single interface having multiple implementations."
- That is making a function or operator to act in different forms depending on the place they are present is called Polymorphism. Overloading is a kind of polymorphism.
- 2 Types of polymorphism:
 1. **Static** : compile time
 2. **Dynamic** : run time

Polymorphism and Overloading

2 Types:

- **Function overloading** which is the process of using the same name for two or more functions.

Refer This Example: <https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/FunctionOverloading/example1.cpp>

- **Operator overloading** which is the process of using the same operator for two or more operands.

Polymorphism and Overloading

Unary Operator Overloading:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/OperatorOverloading/Unary.cpp>

Binary Operator Overloading:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/OperatorOverloading/binary.c>

Polymorphism and Overloading

This refers to the entity which changes its form depending on circumstances at runtime.

Virtual Function:

A virtual function can be defined as the member function within a base class which you expect to redefine in derived classes.

For creating a virtual function, you have to precede your function's declaration within the base class with a **virtual** keyword.

Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Function/virtualfunction.cpp>

What is Pure Virtual Function

Pure Virtual Function is a Virtual function with no body.

```
class classname //This denotes the base class of C++ virtual function
{
public:
    virtual void virtualfunctionname() = 0      //This denotes the pure
                                                virtual function in C++
};
```

Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Function/pureVirtualFunction.cpp>

Abstract classes

- An abstract class is a class that is designed to be specifically used as a base class.
- An abstract class contains at least one pure virtual function.
- You declare a pure virtual function by using a pure specifier (= 0) in the declaration of a virtual member function in the class declaration.
- You cannot use an abstract class as a parameter type, a function return type, or the type of an explicit conversion, nor can you declare an object of an abstract class.

Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c%2B%2BOOPS/Abstract%20Class/abstract.cpp>

Static class

- We can define class members static using **static** keyword. When we declare a member of a class as static it means no matter how many objects of the class are created, there is only **one copy of the static member**.
- A static member is **shared by all objects of the class**. All static data is initialized to zero when the first object is created, if no other initialization is present. We can't put it in the class definition but it can be initialized outside the class as done in the following example by redeclaring the static variable, using the scope resolution operator :: to identify which class it belongs to.

Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c++/OOPS/static/static.c>

Static Function Members:

- By declaring a function member as static, you make it independent of any particular object of the class. A static member function can be called even if no objects of the class exist and the static functions are accessed using only the class name and the scope resolution operator ::.
- A static member function can only access static data member, other static member functions and any other functions from outside the class.
- Static member functions have a class scope and they do not have access to the this pointer of the class. You could use a static member function to determine whether some objects of the class have been created or not.

Refer This Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/c++/OOPS/static/staticFunction.c>

[Module 4]

HTML And CSS

HTML

- HTML (Hypertext Markup Language) is the language used to create web page documents.
- The updated version, XHTML (extensible HTML) is essentially the same language with stricter syntax rules.
- (X)HTML is not a programming language; it is a markup language, which means it is a system for identifying and describing the various components of a document such as headings, paragraphs, and lists.
- You don't need programming skills—only patience and common sense—to write (X)HTML.

HTML Structure

- HTML : HTML stands for Hypertext Markup Language. It is a standard markup language for web page creation. It allows the creation and structure of sections, paragraphs, and links using HTML elements (the building blocks of a web page) such as tags and attributes

HTML Page Structure

```
<!DOCTYPE html>           ← Tells version of HTML
<html>                  ← HTML Root Element
  <head>                ← Used to contain page HTML metadata
    <title>Page Title</title>   ← Title of HTML page
  </head>
  <body>                ← Hold content of HTML
    <h2>Heading Content</h2>   ← HTML heading tag
    <p>Paragraph Content</p>   ← HTML paragraph tag
  </body>
</html>
```

HTML

HTML Elements:

- HTML documents are text files made up of HTML elements. HTML elements are defined using HTML tags.

HTML Tags:

- HTML tags are used to mark-up HTML elements
- HTML tags are surrounded by the two characters < and > The surrounding characters are called angle brackets HTML tags normally come in pairs like and
- The first tag in a pair is the start tag, the second tag is the end tag The text between the start and end tags is the element content HTML tags are not case sensitive, means the same as
- The browser interprets the HTML tags in the source code and displays your content according to those tags.

Basic Elements of HTML

- HTML Basic Tags:

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Defines information about the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment

HTML

Heading Tag Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/HTML/heading.html>

Paragraph Tag Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/HTML/paragraph.html>

Line Break Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/HTML/LineBreak.html>

Horizontal Line Example:

https://github.com/TopsCode/Software-Engineering/blob/master/HTML/horizontTOaPSITLECiHnNOeLOG.IEhS_PtVTm. LDI.

HTML

- HTML : HTML stands for Hypertext Markup Language. It is a standard markup language for web page creation. It allows the creation and structure of sections, paragraphs, and links using HTML elements (the building blocks of a web page) such as tags and attributes

- **HTML Link Tags**

< a href="<http://www.TOPS-int.com>">Go to TOPS-int.com

- **HTML Image**

<https://github.com/TopsCode/Software-Engineering/blob/master/HTML/image1.html>

- **HTML List**

<https://github.com/TopsCode/Software-Engineering/blob/master/HTML/>

- **HTML Table**

<https://github.com/TopsCode/Software-Engineering/blob/master/HTML/Table/c colspan.html>

- **HTML Form**

<https://github.com/TopsCode/Software-Engineering/blob/master/HTML/Form/example1.html>

HTML

- **Formatting Text:**

1. Bold Text: `` and ``
2. Italic Text: `<i>` and ``
3. Marked Formatting: `<mark>`
4. Underlined Text: `<u>` and `<ins>`
5. Strike Text: `<strike>` and ``
6. Monospaced Font: `<tt>`
7. Superscript Text: `<sup>`
8. Subscript Text: `<sub>`
9. Larger Text: `<big>`
10. Smaller Text: `<small>`

Basic Elements of HTML

- **HTML Table Elements:**

```
<table>
<tr>
<th>
<td>
<caption>
<tbody>
<thead>
<tfooter>
```

Attributes:

rowspan
colspan

Basic Elements of HTML

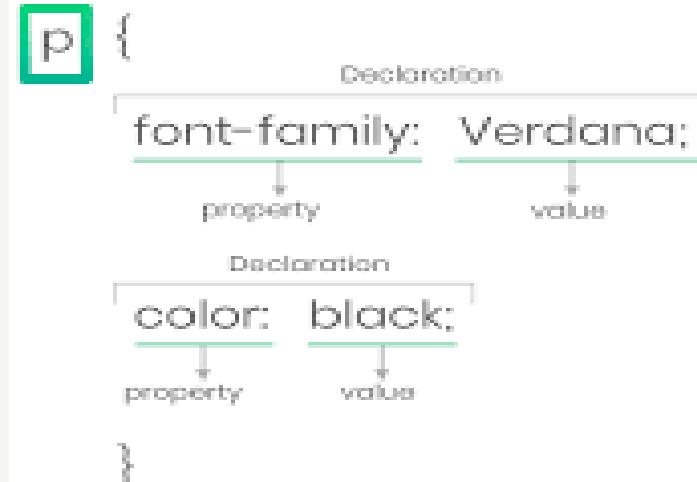
- **HTML FORM Elements:**

- <form>
- <input>
- <textarea>
- <label>
- <fieldset>
- <legend>

CSS

- CSS: CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colors, layout, and fonts, thus making our web pages presentable to the users.
- CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:
 - i. Cascading: Falling of Styles
 - ii. Style: Adding designs/Styling our HTML tags
 - iii. Sheets: Writing our style in different documents

CSS Syntax



CSS

- There are three ways to insert CSS in HTML documents.

1. Inline CSS

exa: <h2 style="color:red;margin-left:40px;">Inline CSS is applied on this heading.</h2>

2. Internal CSS

exa: <style>
body {
background-color: linen;
}
h1 {
color: red; margin-left: 80px;
}
</style>

CSS

3. External CSS

exa:<head>

```
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>
```

- Create Basic layout structure using div tag

CSS

Types of Selectors :

1. Universal Selector
2. HTML selector
3. Id Selector
4. Class Selector
5. Descendant Selector

CSS

Examples :

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/htmlSelector.html>

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/universalSelector.html>

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/classSelector.html>

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/dependantClassSelector.html>

[https://github.com/TopsCode/Software- Engineering/blob/master/CSS/idSelector.html](https://github.com/TopsCode/Software-Engineering/blob/master/CSS/idSelector.html)

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/descendantSelector.css>

CSS

Font Formatting with CSS

1. **font-family**

- The font family of a text is set with the font-family property.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.
- Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Example: <https://github.com/TopsCode/Software-Engineering/blob/master/CSS/Font/font-family.html>

CSS

2. **Serif**

Examples: Times, Times New Roman, Georgia

Serif typefaces have decorative serifs, or slab-like appendages, on the ends of certain letter strokes.

3. **sans-serif**

Examples: Arial, Arial Black, Verdana, Trebuchet MS, Helvetica, Geneva

Sans-serif typefaces have straight letter strokes that do not end in serifs.

They are generally considered easier to read on computer.

4. **Monospace**

Examples: Courier, Courier New, and Andale Mono

In monospace (also called constant width) typefaces, all characters take up the same amount of

4. **Cursive**

Examples: Apple Chancery, Zapf-Chancery, and Comic Sans

Cursive fonts emulate a script or handwritten appearance.

These are rarely specified for professional web pages.

CSS

2. Fantasy

Examples: Impact, Western, or other decorative font

Fantasy fonts are purely decorative and would be appropriate for headlines and other display type.

Fantasy fonts are rarely used for web text due to cross-platform availability and legibility.

Specifying font size:

Use the aptly-named font-size property to specify the size of the text.

font-size

Values: length unit, percentage, xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger | inherit

Default: medium Applies to: all elements Inherits: yes

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/Font/font-size.html>

CSS Box Model

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent
- **Border** - A border that goes around the padding and content. The border is affected by the background color of the box
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box
- **Content** - The content of the box, where text and images appear
In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/boxModel.html>

<https://github.com/TopsCode/Software-Engineering/blob/master/CSS/boxmodel2.html>

[Module 5]

Database

DBMS

- DBMS stands for Data Base Management System.
- Data + Management System
- Database is a collection of inter-related data and Management System is a set of programs to store and retrieve those data.
- DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.
- For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

DBMS

Here is a list of some popular database.

- MySQL
- Microsoft Access
- Oracle
- PostgreSQL
- dBASE
- FoxPro
- SQLite
- IBM DB2
- LibreOffice Base

Need of DBMS

- Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: Storage of data and retrieval of data
Storage:
 - According to the principles of database systems, the data is stored in such a way that it acquires lot less space as the redundant data (duplicate data) has been removed before storage.
 - **Fast Retrieval of data:** Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

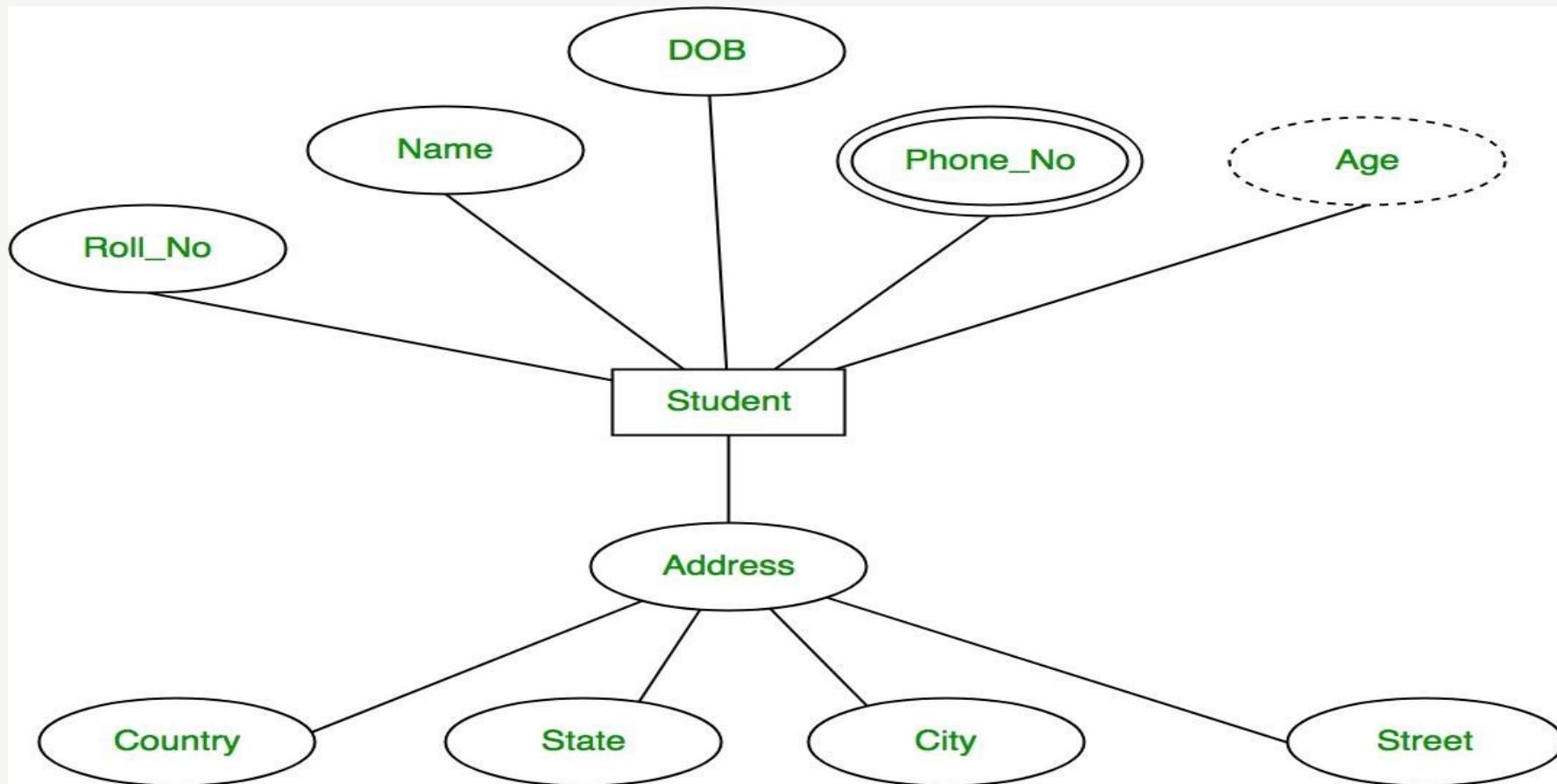
RDBMS

- Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: Storage of data and retrieval of data
 - Storage:
 - According to the principles of database systems, the data is stored in such a way that it acquires lot less space as the redundant data (duplicate data) has been removed before storage.
 - Fast Retrieval of data: Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

E-R Model

- The ER or (Entity Relational Model) is a high-level conceptual data model diagram.
- Entity-Relation model is based on the notion of real-world entities and the relationship between them.
- ER modeling helps you to analyze data requirements systematically to produce a well-designed database.
- So, it is considered a best practice to complete ER modeling before implementing your database.
- Entity relationship diagram displays the relationships of entity set stored in a database.
- In other words, we can say that ER diagrams help you to explain the logical structure of databases.
- At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

E-R Model



Algebra

- Relational Algebra is procedural query language, which takes Relation as input and generate relation as output. Relational algebra mainly provides theoretical foundation for relational databases and SQL.

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π) RENAME (symbol: \bowtie)

Relational Algebra Operations From Set Theory

- UNION (u) INTERSECTION (), DIFFERENCE (-) CARTESIAN PRODUCT (x)

Binary Relational Operations

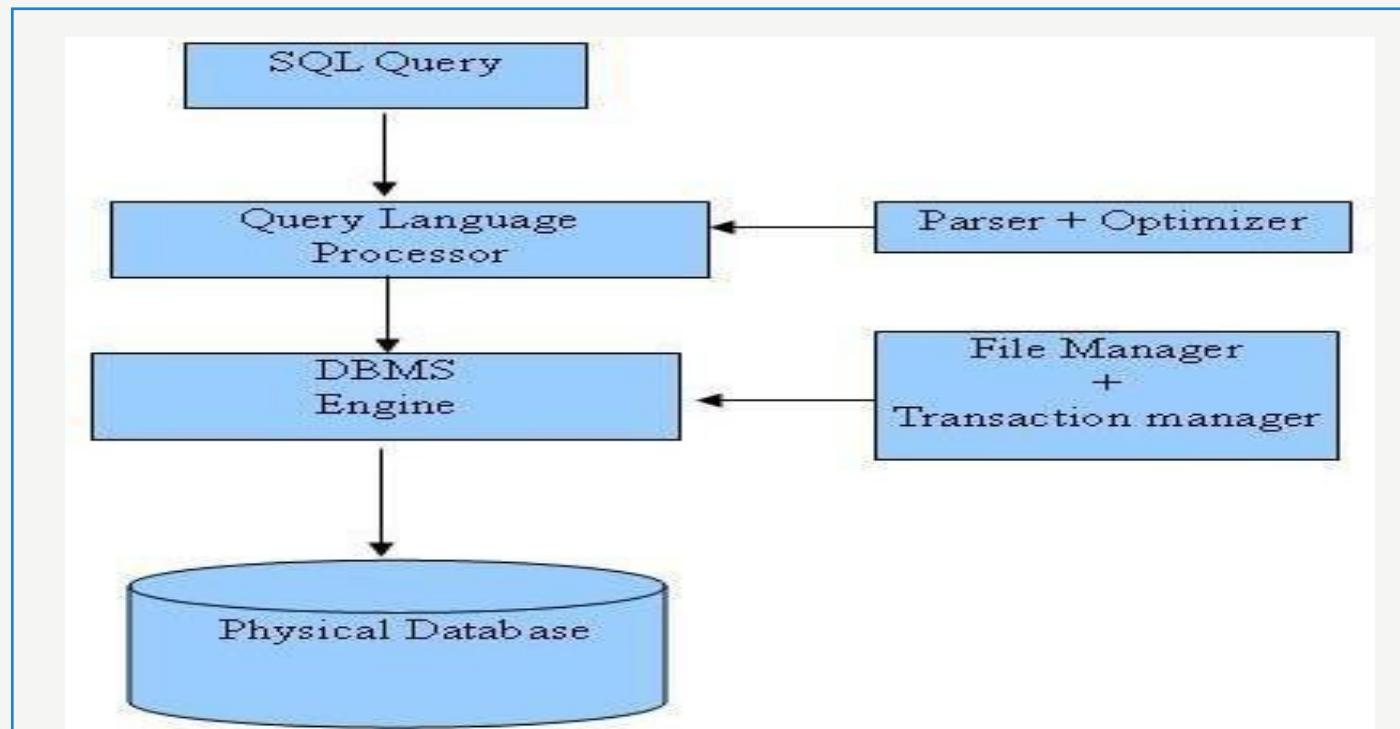
- JOIN DIVISION

What is SQL?

- SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.
- SQL is a language of database, it includes database creation, deletion, fetching rows and modifying rows etc.
- SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL as standard database language.
- Also, they are using different dialects, such as:
- MS SQL Server using T-SQL, ANSI SQL
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc

Objectives

- “The large majority of today's business applications revolve around relational databases and the SQL programming language (Structured Query Language). Few businesses could function without these technologies...”



What is SQL?

- Allows users to access data in relational database management systems. Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database. Allows users to set permissions on tables, procedures, and views
- SQL stands for Structured Query Language
- SQL allows you to access a database
- SQL is an ANSI standard computer language
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert new records in a database

What is SQL?

- SQL can delete records from a database
- SQL can update records in a database
- SQL is easy to learn
- action queries insert, update & delete data
- select queries retrieve data from DB

Constraints(Keys)

Primary Key:

- A primary key is a column of table which uniquely identifies each tuple (row) in that table.
- Primary key enforces integrity constraints to the table.
- Only one primary key is allowed to use in a table.
- The primary key does not accept the any duplicate and NULL values.
- The primary key value in a table changes very rarely so it is chosen with care where the changes can occur in a seldom manner.
- A primary key of one table can be referenced by foreign key of another table.

Constraints(Keys)

Unique Key:

- Unique key constraints also identifies an individual table uniquely in a relation or table.
- A table can have more than one unique key unlike primary key.
- Unique key constraints can accept only one NULL value for column.
- Unique constraints are also referenced by the foreign key of another table.

Constraints(Keys)

Foreign Key:

- When, "one" table's primary key field is added to a related "many" table in order to create the common field which relates the two tables, it is called a foreign key in the "many" table.
- In the example given below, salary of an employee is stored in salary table.
- Relation is established via is stored in "Employee" table. To identify the salary of "Jforeign key column "Employee_ID_Ref" which refers "Employee_ID" field in Employee table.
- of "Jhon" is stored in "Salary" table. But his employee info
- For example, salary hon", his "employee id" is stored with each salary record.

Table : Employee		Table : Salary			
Employee_ID	Employee_Name	Employee_ID_Ref	Year	Month	Salary
1	Jhon	1	2012	April	30000
2	Alex	1	2012	May	31000
3	James	1	2012	June	32000
4	Roy	2	2012	April	40000
5	Kay	2	2012	May	41000
		2	2012	June	42000

Database Normalization

- Normalization is the process of minimizing redundancy (duplicity) from a relation or set of relations.
- Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations.
- Most Commonly used normal forms:

First Normal Form:

- First normal form(1NF) Second normal form(2NF) Third normal form(3NF) Boyce & Code normal form (BCNF)
- If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute.
- A relation is in first normal form if every attribute in that relation is singled valued attribute.

Database Normalization

Second Normal Form:

- To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency.
- relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.
- Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

Database Normalization

Third Normal Form:

- A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.
- A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency $X \rightarrow Y$
 - X is a super key.
 - Y is a prime attribute (each element of Y is part of some candidate key).
- Transitive dependency – If $A \rightarrow B$ and $B \rightarrow C$ are two FDs then $A \rightarrow C$ is called transitive dependency.

SQL Statement Types

- DDL – Data Definition Language
- DML – Data Manipulation Language
- DCL – Data Control Language
- DQL – Data Query Language
- **SQL Join Types**
- INNER JOIN: returns rows when there is a match in both tables.
- LEFT JOIN: returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN: returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN: returns rows when there is a match in one of the tables.

SQL Statement Types

DDL - Data Definition Language

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

SQL Statement Types

DQL – Data Query Language

Command	Description
SELECT	Retrieves certain records from one or more tables

- **DML – Data Manipulation Language**

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

SQL Statement Types

DCL – Data Control Language

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

SQL Statement Types

SQL CREATE DATABASE STATEMENT

```
CREATE DATABASE database_name;
```

SQL DROP DATABASE Statement:

```
DROP DATABASE database_name;
```

SQL USE STATEMENT

```
USE DATABASE database_name;
```

SQL CREATE TABLE STATEMENT

```
CREATE TABLE table_name( column1 datatype, column2 datatype, column3 datatype,  
..... , columnN datatype, PRIMARY KEY( one or more columns ) );
```

SQL Statement Types

SQL DROP TABLE STATEMENT

```
DROP TABLE table_name;
```

SQL TRUNCATE TABLE STATEMENT

```
TRUNCATE TABLE table_name;
```

SQL ALTER TABLE STATEMENT (RENAME)

```
ALTER TABLE table_name RENAME TO new_table_name;
```

SQL INSERT INTO STATEMENT

```
INSERT INTO table_name( column1, column2....columnN) VALUES ( value1, value2.  
valueN);
```

SQL Statement Types

SQL UPDATE STATEMENT

```
UPDATE table_name SET column1 = value1, column2 = value2, columnN=valueN  
[ WHERE CONDITION ];
```

SQL DELETE STATEMENT

```
DELETE FROM table_name WHERE {CONDITION}
```

SQL SELECT STATEMENT

```
SELECT column1, column2....columnN FROM table_name;
```

SQL DISTINCT CLAUSE

```
SELECT DISTINCT column1, column2....columnN FROM table_name;
```

SQL Statement Types

SQL WHERE CLAUSE

SELECT column1, column2....columnN FROM table_name WHERE CONDITION;

SQL AND/OR CLAUSE

SELECT column1, column2....columnN FROM table_name WHERE CONDITION-1
{AND|OR} CONDITION-2;

SQL IN CLAUSE

SELECT column1, column2. column table_name WHERE column_name IN
FROM (val-1, val-2, val-N);

SQL BETWEEN CLAUSE

SELECT column1, column2. column table_name WHERE column_name
FROM BETWEEN val-1 AND val-2;

SQL Statement Types

SQL LIKE CLAUSE

```
SELECT column1, column2.    column table_name WHERE column_name LIKE {  
FROM PATTERN };
```

SQL ORDER BY CLAUSE

```
SELECT column1, column2.    columnN table_name WHERE CONDITION ORDER  
FROM BY column_name {ASC|DESC};
```

SQL GROUP BY CLAUSE

```
SELECT SUM(column_name) table_name WHERE CONDITION GROUP BY  
FROM column_name;
```

SQL COUNT CLAUSE

```
SELECT COUNT(column_name)FROM table_name WHERE CONDITION;
```

SQL Statement Types

SQL HAVING CLAUSE

SELECT SUM(column_name) FROM table_name WHERE CONDITION GROUP BY column_name HAVING (arithematicfunction condition)

SQL CREATE INDEX Statement :

CREATE UNIQUE INDEX index_name ON table_name(column1, column2,...columnN);

SQL DROP INDEX STATEMENT

ALTER TABLE table_name DROP INDEX index_name;

SQL DESC Statement :

DESC table_name;

SQL COMMIT STATEMENT

COMMIT

SQL Statement Types

SQL ROLLBACK STATEMENT

ROLLBACK;

JOIN

A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them.

Different types of Joins are:

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL JOIN

SQL Statement Types

- **1. Inner Join**

The most frequently used and important of the joins is the INNER JOIN. They are also referred to as an EQUIJOIN.

- The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. T
- he query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

Syntax :

```
SELECT table1.column1, table2.column2...FROM table1INNER JOIN table2ON  
table1.common_filed = table2.common_field
```

SQL Statement Types

- **Left Join**
- The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in right table, the join will still return a row in the result, but with NULL in each column from right table.
- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.
- **Syntax:**
- `SELECT table1.column1, table2.column2...FROM table1LEFT JOIN table2ON
table1.common_field = table2.common_field;`

SQL Statement Types

- **Right Join**
- The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in left table, the join will still return a row in the result, but with NULL in each column from left table.
- This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.
- **Syntax:**
- `SELECT table1.column1, table2.column2...FROM table1RIGHT JOIN table2ON table1.common_filed = table2.common_field`

SQL Statement Types

- **Full Join**
- The SQL FULL JOIN combines the results of both left and right outer joins.
- The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.
- **Syntax:**
- `SELECT table1.column1, table2.column2...FROM table1FULL JOIN table2ON
table1.common_filed = table2.common_field;`

SQL Statement Types

Function

- SQL has many built-in functions for performing calculations on data. They are divided into 2 categories:
 1. Aggregate Function
 2. Scalar Function

1. Aggregate Function

- These functions are used to do operations from the values of the column and a single value is returned.
- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

SQL Statement Types

1. AVG():

Syntax: `SELECT AVG(column_name) FROM table_name;`

Example:

```
SELECT AVG(AGE) AS AvgAge FROM Students;
```

2. COUNT()

Syntax: `SELECT COUNT(column_name) FROM table_name;`

Example: `SELECT COUNT(*) AS NumStudents FROM Stuents;`

3. FIRST()

Syntax: `SELECT FIRST(column_name) FROM table_name`

Example:

```
SELECT FIRST(MARKS) AS MarksFirst FROM Students;
```

SQL Statement Types

4. LAST()

Syntax: SELECT LAST(column_name) FROM table_name;

Example: SELECT LAST(MARKS) AS MarksLast FROM Students;

5. MAX()

Syntax: SELECT MAX(column_name) FROM table_name;

Example :SELECT MAX(MARKS) AS MaxMarks FROM Students;

6. MIN(): Similar to Max() we can use MIN() function.

7. SUM() :

Syntax: SELECT SUM(column_name) FROM table_name;

Example: SELECT SUM(MARKS) AS TotalMarks FROM Stuents;

SQL Statement Types

PROCEDURE

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.
- To create procedure, use syntax:

```
CREATE PROCEDURE procedure_name AS sql_statement GO;
```

- To execute created procedure, use syntax:

```
EXEC procedure_name;
```

SQL Statement Types

Transaction Control

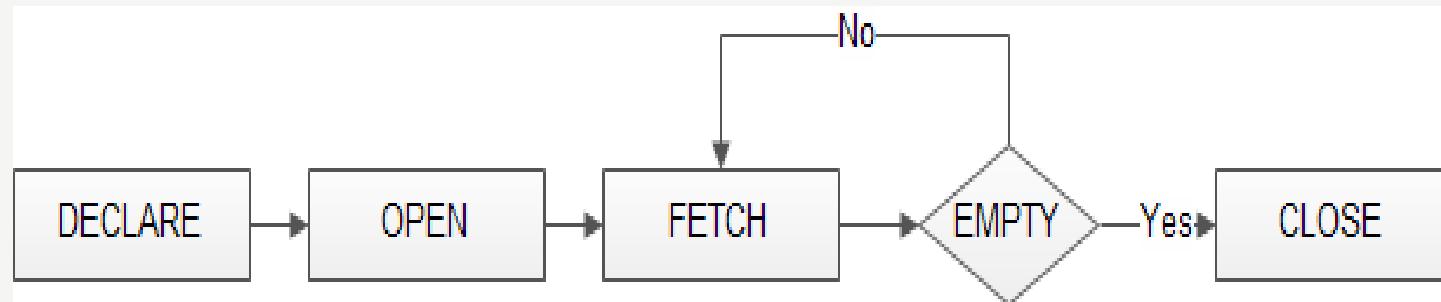
- **The following commands are used to control transactions**
 1. COMMIT – to save the changes.
 2. ROLLBACK – to roll back the changes.
 3. SAVEPOINT – creates points within the groups of transactions in which to ROLLBACK.
- **Commit:**
 - The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.
 - The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

SQL Statement Types

- **Rollback:**
 - The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.
 - This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.
 - The syntax for a ROLLBACK command is as follows – ROLLBACK;
- **Savepoint:**
 - A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.
 - The syntax for a SAVEPOINT command is as shown below.
 - SAVEPOINT SAVEPOINT_NAME;
 - This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.

SQL Statement Types

- **Cursor:**
- It is a temporary area for work in memory system while the execution of a statement is done.
- A Cursor in SQL is an arrangement of rows together with a pointer that recognizes a present row.
- It is a database object to recover information from a result set one row at once.
- It is helpful when we need to control the record of a table in a singleton technique, at the end of the day one row at any given moment. The arrangement of columns the cursor holds is known as the dynamic set.



SQL Statement Types

- **Cursor:**

Syntax:

```
DECLARE variables;  
records;  
create a cursor;  
BEGIN  
OPEN cursor;  
FETCH cursor;  
process the records;  
CLOSE cursor;  
END;
```

SQL Statement Types

Cursor Example:

<https://github.com/TopsCode/Software-Engineering/blob/master/SQL/Cursor/example>

Stored Procedure with one parameter:

<https://github.com/TopsCode/Software-Engineering/tree/master/SQL/Cursor>

Stored Procedure with multiple parameter:

<https://github.com/TopsCode/Software-Engineering/blob/master/SQL/Cursor/multipleParam>

SQL Statement Types

Trigger

- A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs
- For example, a trigger can be invoked when a row is inserted into a specified table.

Syntax:

```
create trigger [trigger_name] [before | after]
{insert | update | delete} on [table_name] [for each row] [trigger_body]
```

Before Trigger:

<https://github.com/TopsCode/Software-Engineering/blob/master/SQL/Trigger/beforeTrigger>

After Trigger:

<https://github.com/TopsCode/Software-Engineering/blob/master/SQL/Trigger/afterTrigger>

11.Tools

Tools can be used in Programming



THANK YOU



TOPS TECHNOLOGIES

Training | Outsourcing | Placement | Study Abroad

Flutter Mobile App Development

Modules

Module 1 - [Flutter Introduction]

Module 2 - [Fundamental-Dart Programming]

Module 3 - [Flutter-UI Designing and Development]

Module 4 - [Flutter- Advance UI Designing and Development]

Module 5 - [Flutter-Offline Database Sqlite]

Module 6 - [Flutter-Advance App Development]

Module 7 - [Flutter-Deployment]

Module 1 : Flutter Introduction



What is ~~Flutter~~ Flutter?

In general, creating a mobile application is a very complex and challenging task. There are many frameworks available, which provide excellent features to develop mobile applications. For developing mobile apps, Android provides a native framework based on Java and Kotlin language, while iOS provides a framework based on Objective-C/Swift language. Thus, we need two different languages and frameworks to develop applications for both OS. Today, to overcome from this complexity, there are several frameworks have introduced that support both OS along with desktop apps. These types of the framework are known as cross-platform development tools.

Why we Are using Flutter ?

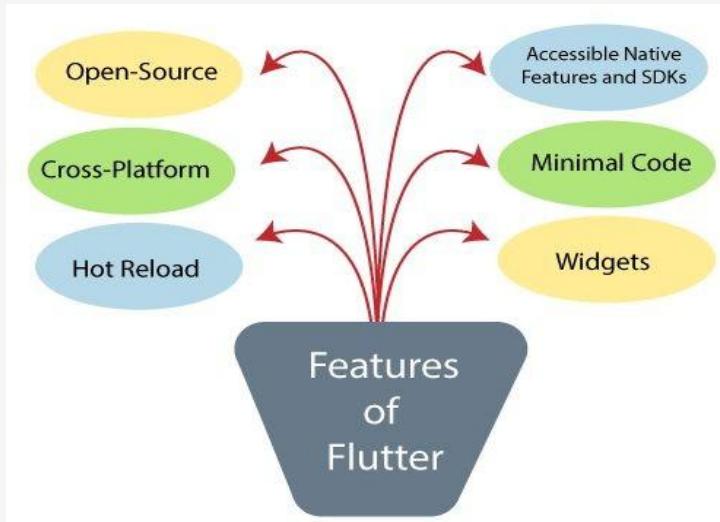
Flutter has been around already since 2015 when Google first introduced it, but the buzz around it has grown stronger only lately. It's a cross-platform tool intended for creating Android and iOS apps from a single code base by using a modern, reactive framework. Flutter apps are built using Dart, a simple object-oriented programming language. The central idea of Flutter revolves around widgets.

- The entire UI is made of combining different widgets, each of which defines a structural element (like a button or menu), a stylistic element (like a font or color scheme), an aspect of layout (like padding), and so on. Flutter does not use OEM widgets, but provides its own ready-made widgets which look native either to Android (Material Design) or iOS apps (Cupertino). It's also possible to create custom widgets

What Makes Flutter Unique?

Flutter is different from other frameworks because it neither uses WebView nor the OEM widgets that shipped with the device. Instead, it uses its own high-performance rendering engine to draw widgets. It also implements most of its systems such as animation, gesture, and widgets in Dart programming language that allows developers to read, change, replace, or remove things easily. It gives excellent control to the developers over the system.

Features Of Flutter



- **Open-Source:** Flutter is a free and open-source framework for developing mobile applications.
- **Cross-platform:** This feature allows Flutter to write the code once, maintain, and can run on different platforms. It saves the time, effort, and money of the developers.
- **Hot Reload:** Whenever the developer makes changes in the code, then these changes can be seen instantaneously with Hot Reload. It means the changes immediately visible in the app itself. It is a very handy feature, which allows the developer to fix the bugs instantly.

- **Minimal code:** Flutter app is developed by Dart programming language, which uses JIT and AOT compilation to improve the overall start-up time, functioning and accelerates the performance. JIT enhances the development system and refreshes the UI without putting extra effort into building a new one.
- **Widgets:** The Flutter framework offers widgets, which are capable of developing customizable specific designs. Most importantly, Flutter has two sets of widgets: Material Design and Cupertino widgets that help to provide a glitch-free experience on all platforms.

- For Flutter You Need to Learn Dart Programming
- So Let's start with Dart Programming it's a very easy and understandable programminglanguage If you are aware from java.



Module 2

Fundamentals Of Dart Programming

Dart SDK and Installation

SDK Stands For Software Development Kit

1) Download Dart SDK From the given link:

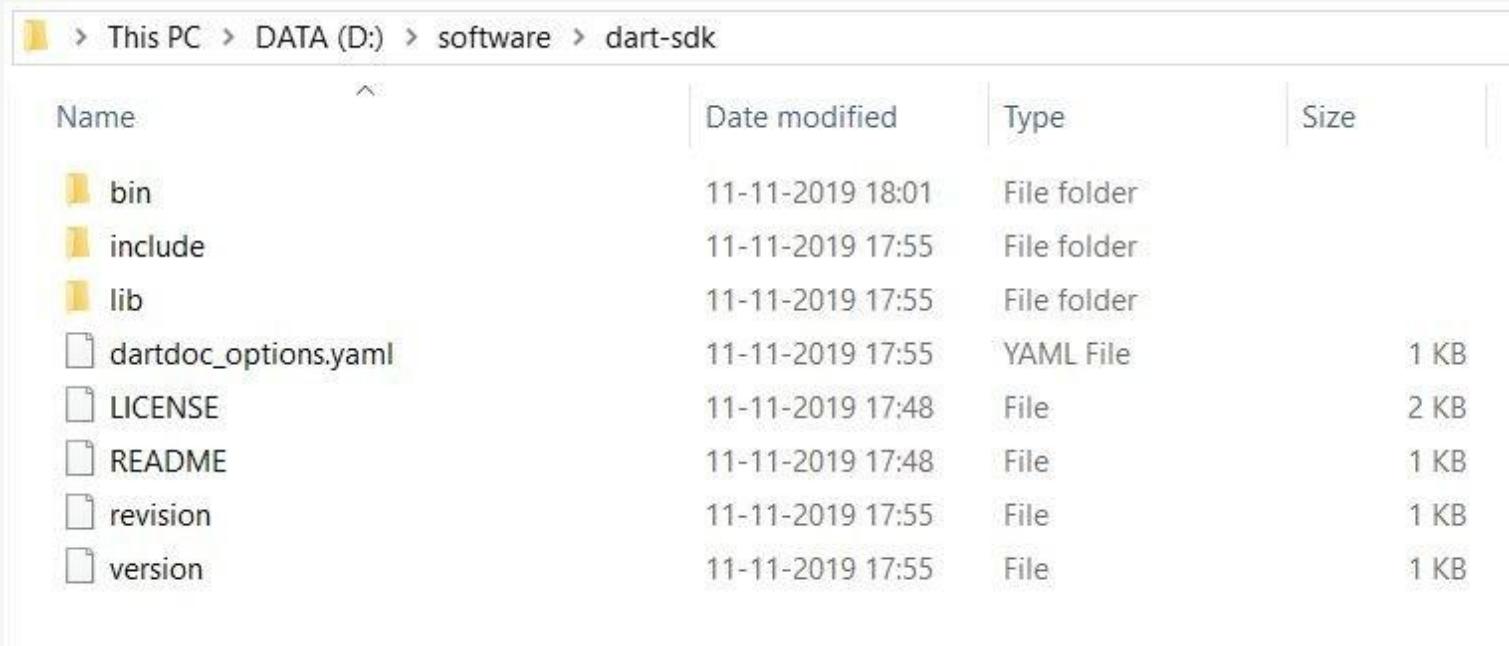
<https://gekorm.com/dart-windows/>

2) Or You can Download the Direct Dart SDK Zip File

<https://dart.dev/get-dart/archive>

3) Just Extract the Zip File and set the Dart SDK Path in Environment Variable

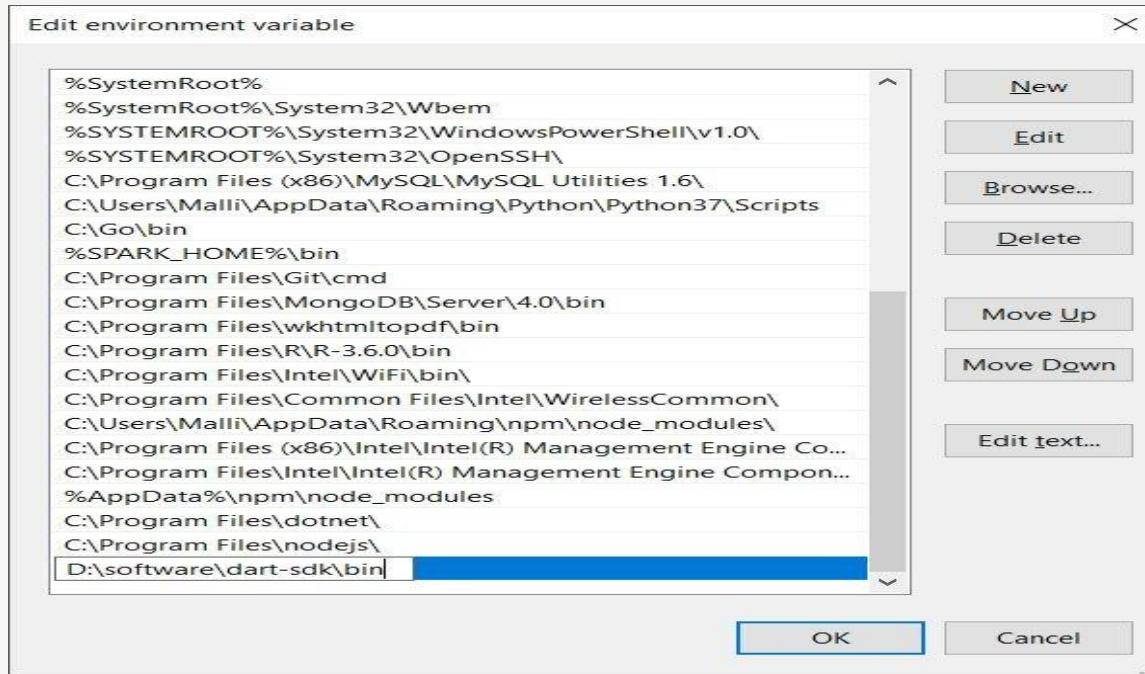
Step - 1 :



The screenshot shows a Windows File Explorer window with the following file structure and details:

Name	Date modified	Type	Size
bin	11-11-2019 18:01	File folder	
include	11-11-2019 17:55	File folder	
lib	11-11-2019 17:55	File folder	
dartdoc_options.yaml	11-11-2019 17:55	YAML File	1 KB
LICENSE	11-11-2019 17:48	File	2 KB
README	11-11-2019 17:48	File	1 KB
revision	11-11-2019 17:55	File	1 KB
version	11-11-2019 17:55	File	1 KB

Step-2:



Step-3:

Open Your CMD and type Dart it will Generate The code

Here,Your Dart Installation Is Over.

- Dart is a Language
- we can use this language for flutter development
- Using Flutter we can create hybrid applications

- Dart is a general-purpose, high-level modern programming language which is originally developed by Google. It is the new programming language which is emerged in 2011, but its stable version was released in June 2017. Dart is not so popular at that time, but It gains popularity when it is used by the Flutter.
- Dart is an open-source programming language which is widely used to develop the mobile application, modern web-applications, desktop application, an the Internet of Things (IoT) using by Flatter framework.
- It also supports a few advance concepts such as interfaces, mixins, abstract classes, redfield generics, and type interface. It is a compiled language and supports two types of compilation techniques.

- AOT (Ahead of Time) - It converts the Dart code in the optimized JavaScript code with the help of the dar2js compiler and runs on all modern web-browser. It compiles the code at build time.
- JOT (Just-In-Time) - It converts the byte code in the machine code (native code), but only code that is necessary.

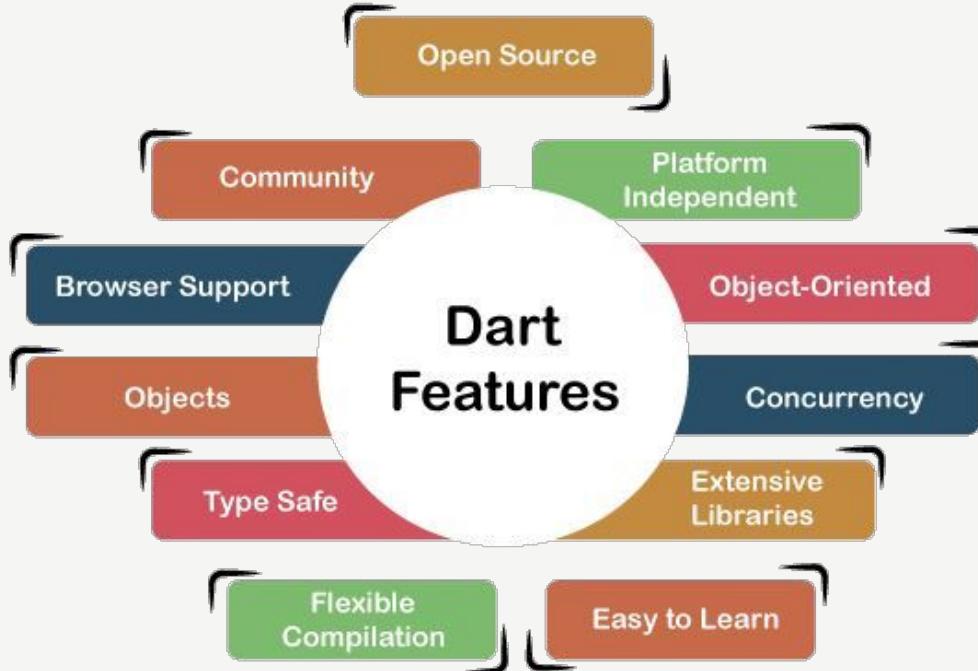
Why Dart?

We define the characteristics of Dart in the following point.

- o Dart is a platform-independent language and supports all operating systems such as Windows, Mac, Linux, etc.
- o It is an open-source language, which means it available free for everyone. It comes with a BSD license and recognized by the ECMA standard.
- o It is an object-oriented programming language and supports all features of oops such as inheritance, interfaces, and optional type features.

Dart Features

The Dart is an object-oriented, open-source programming language which contains many useful features. It is the new programming language and supports an extensive range of programming utilities such as interface, collections, classes, dynamic and optional typing. It is developed for the server as well as the browser. Below is the list of the important Dart features.



Dart Data types

- Number
- Strings
- Boolean
- Lists
- Maps
- Runes
- Symbols

Dart Variable Declaration

- Dart supports type-checking, it means that it checks whether the data type and the data that
- variable holds are specific to that data or not.

```
int value=20;
```

```
var value=20;
```

```
String name="Tops Technologies" var  
name="Tops Technologies"
```

String Interpolation

String interpolation is the process of inserting variable values into placeholders in a string literal.

We use \$ symbol for String interpolation.

https://github.com/TopsCode/Flutter/blob/main/Module%202/1_hello_world.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/2_builtin_data_types.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/3string_and_string_interpolation.dart

Constants

Constants are objects whose values cannot be changed during the execution of the program. Hence, they are a type of immutable object. A constant cannot be reassigned any value if a value has been assigned to it earlier. If we try to reassign any value to a constant, Dart throws an exception

- **without data type**

```
final variable_name;
```

- **with data type**

```
final data_type    variable_name;
```

<https://github.com/TopsCode/Flutter/blob/main/Module%202/4constants.dart>

Operators

1. Arithmetic Operators
2. Relational Operators
3. Type Test Operators
4. Bitwise Operators
5. Assignment Operators
6. Logical Operators
7. Conditional Operator
8. Cascade Notation
Operator

https://github.com/TopsCode/Flutter/blob/main/Module%202/5Arithmetic_operator.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/6Relational_operator.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/7Type_Test.dart

<https://github.com/TopsCode/Flutter/blob/main/Module%202/8Bitwise.dart>

<https://github.com/TopsCode/Flutter/blob/main/Module%202/9Assignment.dart>

<https://github.com/TopsCode/Flutter/blob/main/Module%202/10Logical.dart>

Conditional Statement

1. Simple If Statement
2. if-else Statement
3. Else if Ladder Statement
4. Nested If Statement
5. Switch case Statement

<https://github.com/TopsCode/Flutter/blob/main/Module%202/11Conditional.dart>

<https://github.com/TopsCode/Flutter/blob/main/Module%202/12Cascade.dart>

https://github.com/TopsCode/Flutter/blob/main/Module%202/13.1_else_if.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/14_conditional_expressions.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/15_switch_and_case.dart

Looping Statement

1. **for loop**
2. **for... in loop**
3. **for each loop**
4. **while loop**
5. **do-while loop**

https://github.com/TopsCode/Flutter/blob/main/Module%202/16_For%20Loop.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/17_For%20in%20Loop.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/18_For%20each%20Loop.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/19_while_loop.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/20_do_while_loop.dart

Jumping Statement

1. Break Statement:

This statement is used to break the flow of control of the loop i.e if it is used within a loop then it will terminate the loop whenever encountered. It will bring the flow of control out of the nearest loop.

2. Continue Statement:

While the **break** is used to end the flow of control, **continue** on the other hand is used to continue the flow of control. When a continue statement is encountered in a loop it doesn't terminate the loop but rather jump the flow to next iteration.

https://github.com/TopsCode/Flutter/blob/main/Module%202/21_break_keyword.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/22_continue_keyword.dart

Functions & Parameters:

Function is a set of statements that take inputs, do some specific computation and produces output. Functions are created when certain statements are repeatedly occurring in the program and a function is created to replace them.

Functions make it easy to divide the complex program into smaller sub-groups and increase the code reusability of the program.

```
return_type function_name ( parameters )  
{  
    // Body of function return value;  
}
```

Another Type of Function is:

- Function without parameter and return value.

There are Three Type Of Parameters

- 1.Optional Parameters
- 2.Named Parameters
- 3.Default Parameters

https://github.com/TopsCode/Flutter/blob/main/Module%202/23_functions.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/24_function_expression.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/25_optional_positional_params.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/26_named_parameters.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/27_default_parameters.dart

Advance Dart Programming

(OOP) Object Oriented Programming:

There are 6 features of Object Oriented Programming

- 1.Object
- 2.Class
- 3.Abstraction
- 4.Encapsulation
- 5.Inheritance
- 6.Polymorphism

Definitions for OOP:

1.Object : Any Entity Which has Own State and Behaviour That is called an Object

2.Class : group of data members & member functions

Like person can be class having data members height and weight and member functions as get_details() and put_details() to manipulate on details

3.Abstraction : Hiding Internal Details and Showing

Functionalities That is called an Abstraction.

4.Encapsulation : Wrapping up or Binding of data that is called

and Encapsulation.

5. Inheritance : When One Object Acquire All The Properties and Behaviour of Parent class That is called an Inheritance.

6. Method Overriding : When we declare the same method in the subclass, which is previously defined in the superclass is known as the method overriding. The subclass can define the same method by providing its own implementation, which is already exists in the superclass. The method in the superclass is called method overridden, and method in the subclass is called method overriding.

Note : Dart doesn't have overloading as it is our experience that overloading leads to confusion. Especially when you intend to override a method but accidentally overload it. We think that optional arguments (named or not) are a better alternative. Overloading is orthogonal to type annotations.

Class

Dart classes are the blueprint of the object, or it can be called object constructors. A class can contain fields, functions, constructors, etc. It is a wrapper that binds/encapsulates the data and functions together; that can be accessed by ~~areat object~~. A class can refer to as user-defined data type which defines characteristics by its all objects.

Syntax:

```
class ClassName  
{  
    <fields>  
    <getters/setters>  
    <constructor>  
    <functions>  
}
```

Object

An object is a real-life entity such as a table, human, car, etc. The object has two characteristics - state and behavior. Let's take an example of a car which has a name, model name, price and behavior moving, stopping, etc. The object-oriented programming offers to identify the state and behavior of the object.

We can access the class properties by creating an object of that class. In Dart, The object can be created by using a new keyword followed by class name.

Creating Class Objects in Dart

After creating the class, we can create an instance or object of that class which we

want to access its fields and functions. The new keyword is used to declare class

followed by the class name. The general syntax of creating an object of a class

Syntax:

```
var objectName = new ClassName(<constructor_arguments>)
```

https://github.com/TopsCode/Flutter/blob/main/Module%202/28_class_and_objects.dart

Constructor

A constructor is a different type of function which is created with same name as its class name. The constructor is used to initialize an object when it is created. When we create the object of class, then constructor is automatically called. It is quite similar to class function but it has no explicit return type. The generative constructor is the most general form of the constructor, which is used to create a new instance of a class.

Types of Constructors

There are three types of constructors in Dart as given below.

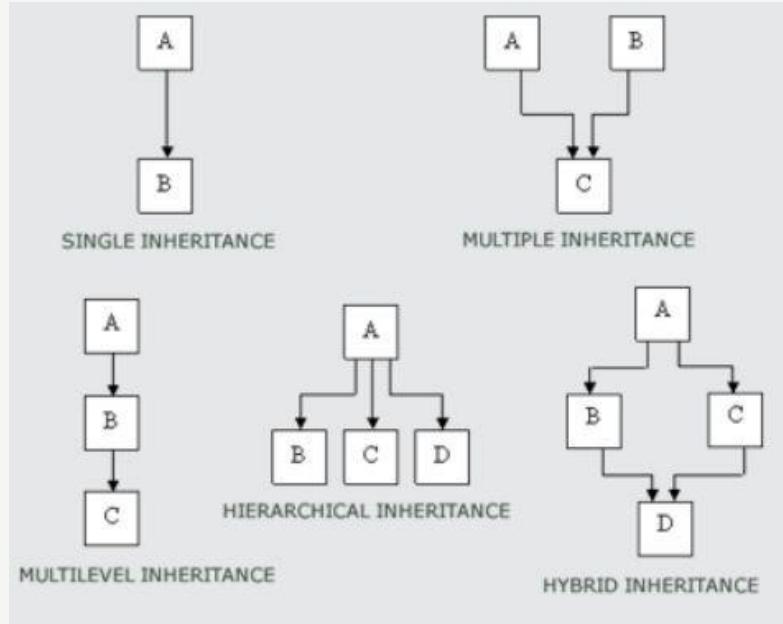
- Default Constructor or no-arg
- Parameterized Constructor
- Named Constructor

https://github.com/TopsCode/Flutter/blob/main/Module%202/29_constructors.dart

Getter-Setter

https://github.com/TopsCode/Flutter/blob/main/Module%202/30_getters_setters.dart

Inheritance



https://github.com/TopsCode/Flutter/blob/main/Module%202/31_inheritance.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/32_inheritance_with_constructors.dart

Interface

An interface defines the syntax that any entity must adhere to. Dart does not have any separate syntax to define interfaces. An Interface defines the same as the class where any set of methods can be accessed by an object. The Class declaration can interface itself.

The keyword **implement** is needed to be writing, followed by class name to be able to use the interface. Implementing class must provide a complete definition of all the functions of the implemented interface. We can say that a class must define every function with the body in the interface that we want to achieve.

Declaring an Interface

Dart doesn't provide syntax for declaring interface directly. Implicitly, a class declaration itself an interface containing the entire instance member of the class and of any interfaces it implements.

Implementing an Interface

To work with interface methods, the interface must be implemented by another class using the implements keyword. A class which is implemented the interface must provide a full implementation of all the methods that belongs to the interface. Following is the syntax of the implementing interface.

```
class ClassName implements InterfaceName
```

https://github.com/TopsCode/Flutter/blob/main/Module%202/33_interface.dart

Method Overriding

https://github.com/TopsCode/Flutter/blob/main/Module%202/34_method_overriding.dart

Abstract Class

Abstract classes are the classes in Dart that has one or more abstract method. Abstraction is a part of the data encapsulation where the actual internal working of the function hides from the users. They interact only with external functionality. We can declare the abstract class by using the abstract keyword. There is a possibility that an abstract class may or may not have abstract methods.

Abstract methods are those methods, which are declared without implementation. The concrete methods or normal methods are declared with implementation. An abstract class can contain both types of methods, but a normal class is not allowed to have abstract methods.

https://github.com/TopsCode/Flutter/blob/main/Module%202/35_abstract_class_method.dart

Keywords

This Keyword:

The this keyword is used to refer the current class object. It indicates the current instance of the class, methods, or constructor. It can be also used to call the current class methods or constructors. It

eliminates the uncertainty between class attributes and the parameter names are the same. If we declare the class attributes same as the parameter name, that situation will create ambiguity in the program, then the this keyword can remove the ambiguity by prefixing the class attributes. It can be passed as an argument in the class method or constructors.

static Keyword:

The this keyword is used to refer the current class object. It indicates the current instance of the class, methods, or constructor. It can be also used to call the current class methods or constructors. It eliminates the uncertainty between class attributes and the parameter names are the same. If we declare the class attributes same as the parameter name, that situation will create ambiguity in the program, then the this keyword can remove the ambiguity by prefixing the class attributes. It can be passed as an argument in the class method or constructors.

Static Keyword

There are two things in Static Keyword

- 1) Static Variable
- 2) Static Method

Super

The super keyword is used to denote the instant parent class object of the current child class. It is used to invoke superclass methods, superclass constructor in its child class. The super keyword's main objective is to remove the confusion between parent class and subclass with the same method name. It is also used to refer to the superclass properties and methods.

Usage of static Keyword

- When parent class and child class have members with the same name, then super keyword can be accessed data members of parent class in child class.
- It is used to access the parent class constructor in the child class.
- Using the super keyword, we can access the superclass method that is overridden by the subclass.

https://github.com/TopsCode/Flutter/blob/main/Module%202/36_this.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/37.1_static.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/37_static_method_variable.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/38_super.dart

Dart Collections

Collections are groups of objects that represent a particular element.

The dart::collection library is used to implement the collection in dart.

There are a variety of collections available in dart.

- 1.List
- 2.Set
- 3.Map

Exception Handling

Dart Exceptions are the run-time error. It is raised when the program gets execution. The program doesn't report the error at compile time when the program runs internally and if Dart compiler found something not appropriate. Then, it reports run-time error and the execution of program is terminated abnormally. This type of error is called Exceptions. For example - A given number is divided by the zero or we try to access the elements from the empty list.

https://github.com/TopsCode/Flutter/blob/main/Module%202/39_list_fixed_length.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/40_list_growable.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/41_maps_and_hashmap.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/42_set_and_hashset.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/43_callable_classes.dart

Async & Await

Async: It simply allows us to write promises based code as if it was synchronous and it checks that we are not breaking the execution thread. It operates asynchronously via the event-loop. Async functions will always return a value. It makes sure that a promise is returned and if it is not returned then JavaScript automatically wraps it in a promise which is resolved with its value.

Await: Await function is used to wait for the promise. It could be used within the async block only. It makes the code wait until the promise returns a result.
It only makes the async block wait.

https://github.com/TopsCode/Flutter/blob/main/Module%202/45_Async_Await.dart

https://github.com/TopsCode/Flutter/blob/main/Module%202/44_Exception.dart

Module 3

UI Designing and Development

Flutter Project Creation

How to Create Project in Flutter with Android Studio?

Step 1: Download Android Studio

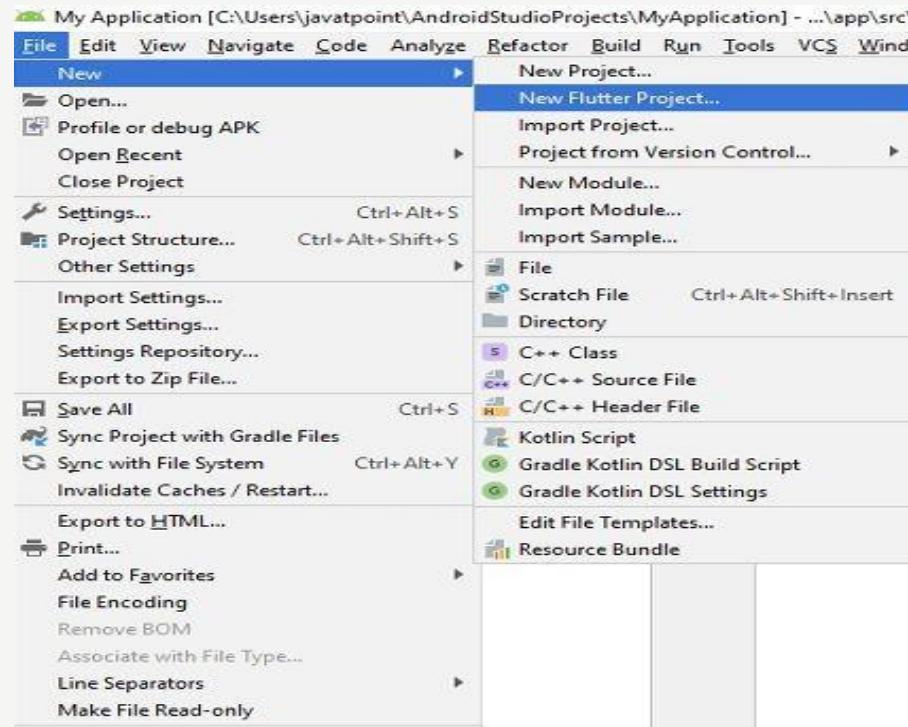
Step 2: Install Two Plugings 1)Dart 2)Flutter

Step 3:Download Flutter SDK from following link

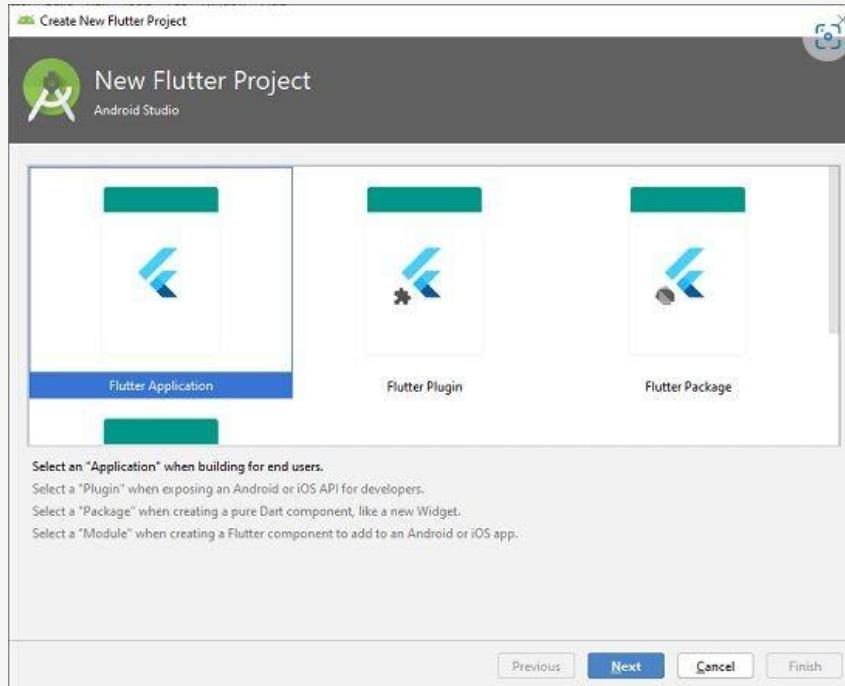
<https://docs.flutter.dev/get-started/install/windows>

Step 4:Create a New Project For Flutter

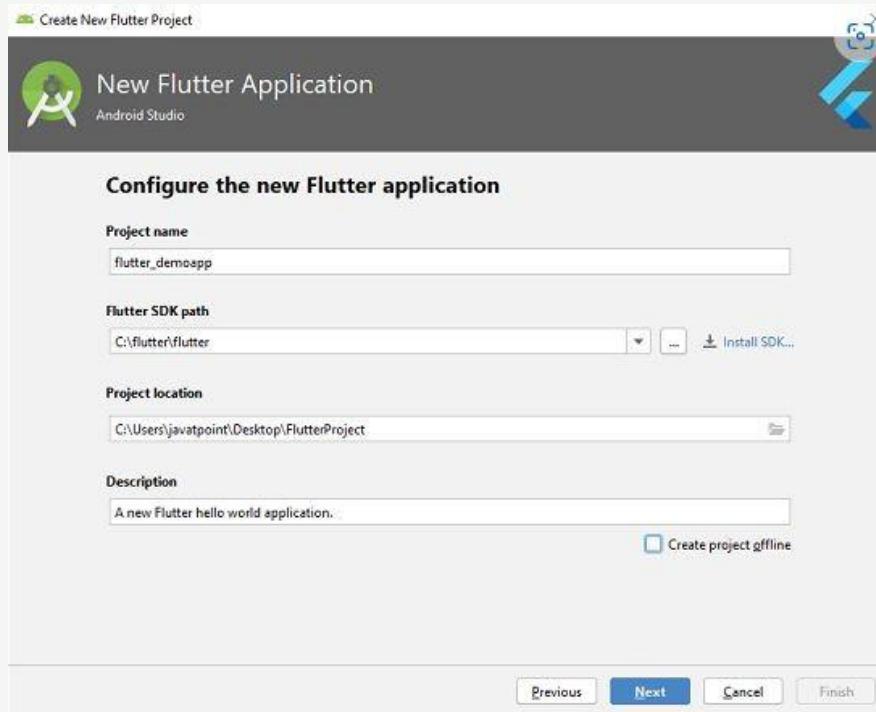
Step 5:



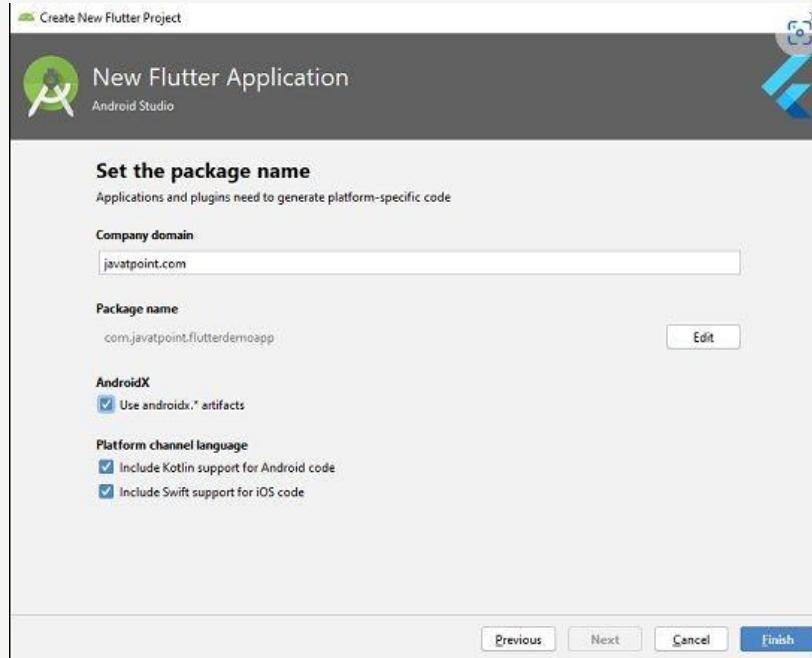
Step 6:



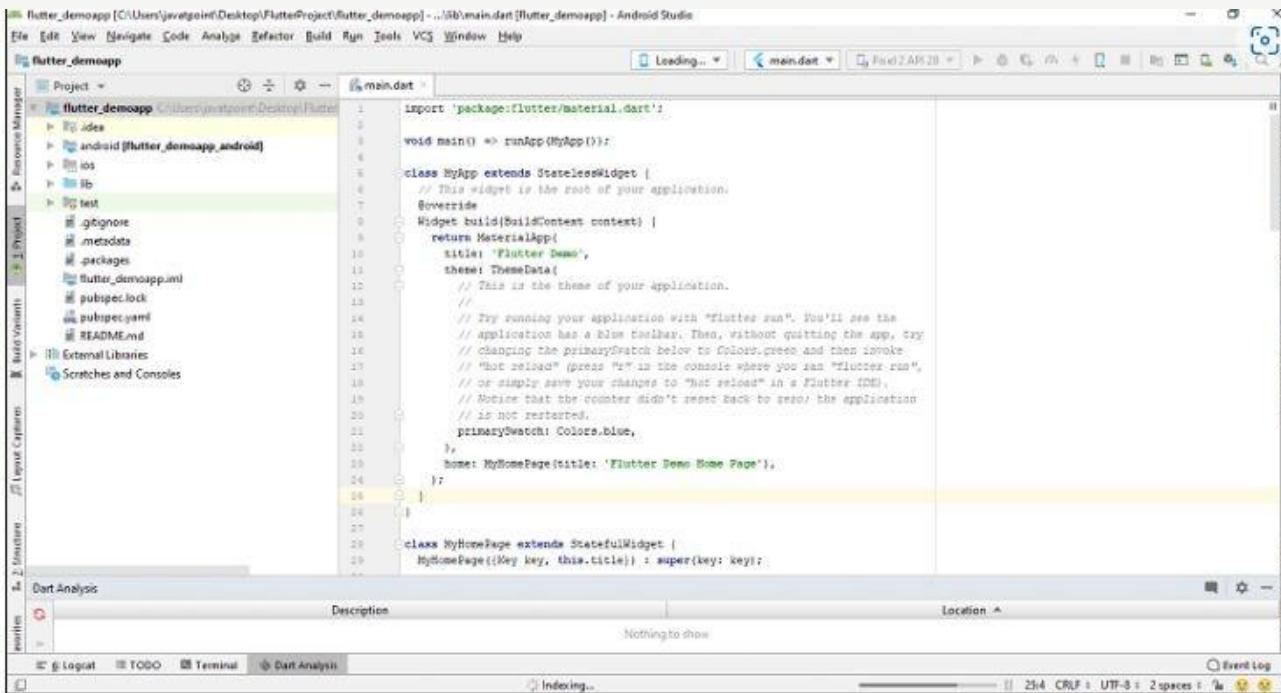
Step 7:



Step 8:



Step 9:



The screenshot shows the Android Studio interface with the project 'flutter_demoapp' open. The main.dart file is the active code editor. The code implements a simple Flutter application structure with a MaterialApp and a HomePage.

```
import 'package:flutter/material.dart';

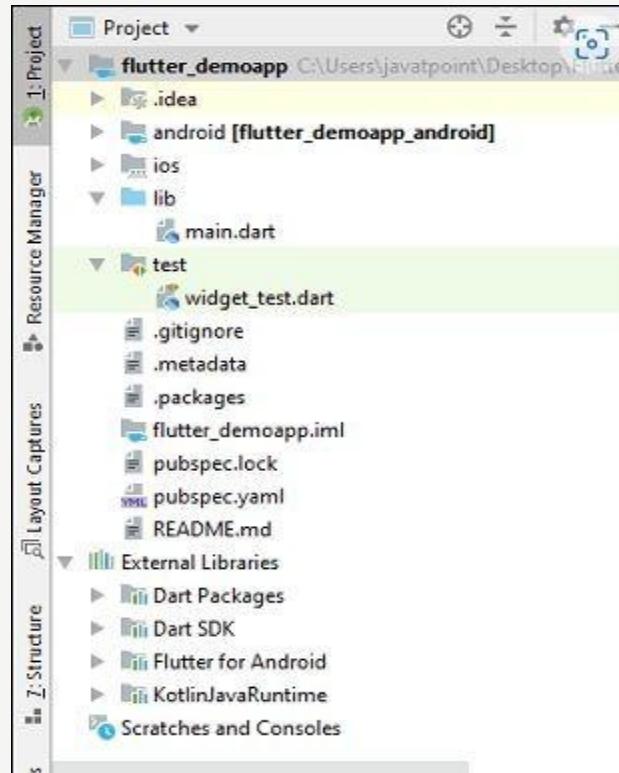
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero? The application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

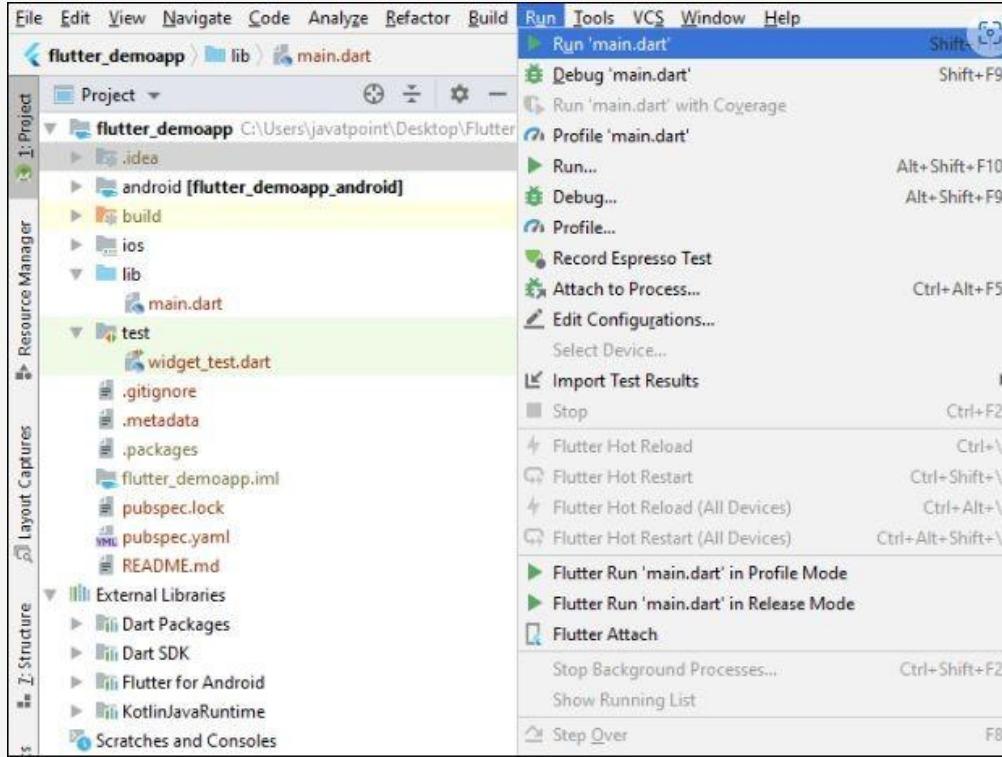
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;
}
```

Step 10:



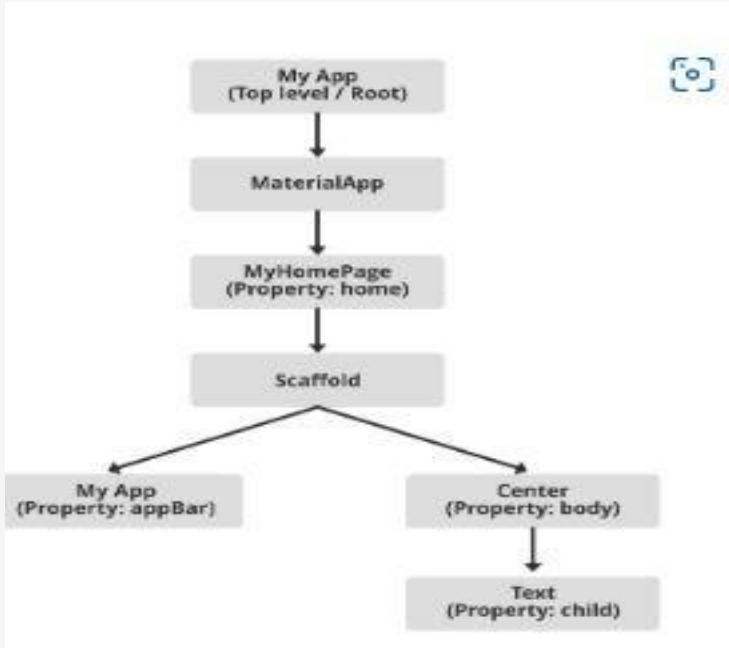
Step 11 :



Step
12:



Flutter Architecture



Flutter architecture application mainly consists of:

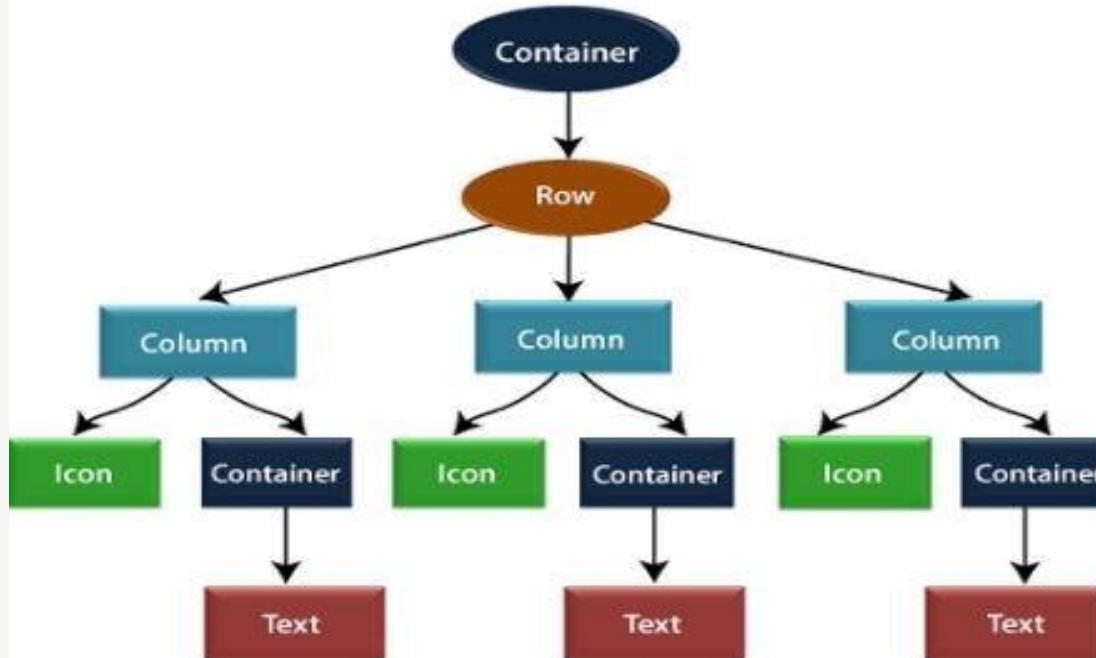
- **Widgets**
- **Gestures**
- **Concept of State**
- **Layers**

Flutter Widgets

Widgets are the primary component of any flutter application. It acts as a UI for the user to interact with the application. Any flutter application is itself a widget that is made up of a combination of widgets. In a standard application, the root defines the structure of the application followed by a `MaterialApp` widget which basically holds its internal components in place. This is where the properties of the UI and the application itself is set. The `MaterialApp` has a `Scaffold` widget that consists of the visible components (widgets) of the application. The `Scaffold` has two primary properties namely the `body` and the `appbar`. It holds all the child widgets and this is where all its properties are defined. The below diagram shows the hierarchy of a flutter application.

Flutter Layouts

The main concept of the layout mechanism is the widget. We know that flutter assume everything as a widget. So the image, icon, text, and even the layout of your app are all widgets. Here, some of the things you do not see on your app UI, such as rows, columns, and grids that arrange, constrain, and align the visible widgets are also the widgets.



Let us learn how we can create and display a simple widget. The following steps show how to layout a widget:

Step 1: First, you need to select a Layout widget.

Step 2: Next, create a visible widget.

Step 3: Then, add the visible widget to the layout widget.

Step 4: Finally, add the layout widget to the page where you want to display.

Types of Layout Widgets :

We can categorize the layout widget into two types:

1. Single Child Widget
2. Multiple Child Widget

Single Child Widget

The single child layout widget is a type of widget, which can have only one child widget inside the parent layout widget. These widgets can also contain special layout functionality. Flutter provides us many single child widgets to make the app UI attractive. If we use these widgets appropriately, it can save our time and makes the app code more readable. The list of different types of single child widgets are:

Container: It is the most popular layout widget that provides customizable options for painting, positioning, and sizing of widgets.

Let us check some of the most important single child layout widgets provided by Flutter –

- Padding – Used to arrange its child widget by the given padding. Here, padding can be provided by EdgeInsets class.
- Align – Align its child widget within itself using the value of alignment property. The value for alignment property can be provided by FractionalOffset class. The FractionalOffset class specifies the offsets in terms of a distance from the top left.

Some of the possible values of offsets are as follows –

- FractionalOffset(1.0, 0.0) represents the top right.
- FractionalOffset(0.0, 1.0) represents the bottom left.
- FittedBox – It scales the child widget and then positions it according to the specified fit.
- AspectRatio – It attempts to size the child widget to the specified aspect ratio
- ConstrainedBox

- Baseline
- FractinallySizedBox
- IntrinsicHeight
- IntrinsicWidth
- LiimitedBox
- OffStage
- OverflowBox
- SizedBox

- SizedOverflowBox
- Transform
- CustomSingleChildLayout

Multiple Child Widgets

In this category, a given widget will have more than one child widgets and the layout of each widget is unique.

For example, Row widget allows the laying out of its children in horizontal direction, whereas Column widget allows laying out of its children in vertical direction. By composing Row and Column, widget with any level of complexity can be built.

Let us learn some of the frequently used widgets in this section.

- Row – Allows to arrange its children in a horizontal manner.
- Column – Allows to arrange its children in a vertical manner.
- ListView – Allows to arrange its children as list.
- GridView – Allows to arrange its children as gallery.
- Expanded – Used to make the children of Row and Column widget to occupy the maximum possible area.
- Table – Table based widget.
- Flow – Flow based widget.
- Stack – Stack based widget.

MaterialApp

MaterialApp Class: MaterialApp is a predefined class in a flutter. It is likely the main or core component of flutter. We can access all the other components and widgets provided by Flutter SDK. Text widget, Dropdownbutton widget, [AppBar](#) widget, [Scaffold](#) widget, [ListView](#), [StatelessWidget](#), [StatefulWidget](#), IconButton widget, TextField widget, Padding widget, ThemeData widget, etc. are the widgets that can be accessed using MaterialApp class. There are many widgets more that are accessed using MaterialApp class. Using this widget, we can make an attractive app.

Scaffold

Scaffold is a class in flutter which provides many widgets or we can say APIs like Drawer, Snackbar, BottomNavigationBar, FloatingActionButton, AppBar, etc. Scaffold will expand or occupy the whole device screen. It will occupy the available space. Scaffold will provide a framework to implement the basic material design layout of the application.

Container

Container class in flutter is a convenience widget that combines common painting, positioning, and sizing of widgets. A Container class can be used to store one or more widgets and position them on the screen according to our convenience. Basically, a container is like a box to store contents. A basic container element that stores a widget has a margin, which separates the present container from other contents. The total container can be given a border of different shapes, for example, rounded rectangles, etc.

A container surrounds its child with padding and then applies additional constraints to the padded extent (incorporating the width and height as constraints, if either is non-null).

<https://github.com/TopsCode/Flutter/tree/main/Module%203/1.%20MaterialApp>

<https://github.com/TopsCode/Flutter/tree/main/Module%203/2.%20Scaffold>

<https://github.com/TopsCode/Flutter/tree/main/Module%203/3.%20Container>

State

The State is the information that can be read synchronously when the widget is built and might change during the lifetime of the widget.

In other words, the state of the widget is the data of the objects that its properties (parameters) are sustaining at the time of its creation
(when

the widget is painted on the screen). The state can also change
when

it is used for example when a CheckBox widget is clicked a
check

appears on the
box.

Stateful Widget

The widgets whose state can be altered once they are built are called stateful Widgets. These states are mutable and can be changed multiple times in their lifetime. This simply means the state of an app can change multiple times with different sets of variables, inputs, data. Below is the basic structure of a stateful widget. Stateful widget overrides the `createState()` and returns a State. It is used when the UI can change dynamically. Some examples can be CheckBox, RadioButton, Form, TextField.

Classes that inherit “Stateful Widget” are immutable. But the State is mutable which changes in the runtime when the user interacts with it.

Stateless Widget

The widgets whose state can not be altered once they are built are called stateless widgets. These widgets are immutable once they are built i.e any amount of change in the variables, icons, buttons, or retrieving data can not change the state of the app. Below is the basic structure of a stateless widget. Stateless widget overrides the build() method and returns a widget. For example, we use Text or the Icon is our flutter application where the state of the widget does not change in the runtime. It is used when the UI depends on the information within the object itself. Other examples can be Text, RaisedButton, IconButton.

Stateful vs. Stateless

Differences Between Stateless and Stateful Widget:

Stateless Widget:

- Stateless Widgets are static widgets.
- They do not depend on any data change or any behavior change.
- Stateless Widgets do not have a state, they will be rendered once and will not update themselves, but will only be updated when external data changes.
- For Example: Text, Icon, *RaisedButton* are Stateless Widgets.

Stateful Widget:

- Stateful Widgets are dynamic widgets.
- They can be updated during runtime based on user action or data change.
- Stateful Widgets have an internal state and can re-render if the input data changes or if Widget's state changes.
- For Example: Checkbox, Radio Button, Slider are Stateful Widgets

Row and Column Widgets

Row and Column are the two most important and powerful widgets in Flutter. These widgets let you align children horizontally and vertically as per the requirement. As we know that when we design any UI(User Interface) in a flutter, we need to arrange its content in the Row and Column manner so these Row and Column widgets are required when designing UI.

Properties of Row and Column

Widgets:

- **children:** This property takes in *List<Widget>*, that is a list of Widgets to display inside the *Row* or the *Column* widget.
- **clipBehaviour:** This property holds Clip class as the object to decide whether the content on the *Row* or *Column* should be clipped or not.

- **crossAxisAlignment:** The crossAxisAlignment takes in CrossAxisAlignment enum as the object to how the children's widgets should be places in crossAxisAlignment. For Row it is vertical and for Column it is horizontal.
- **direction:** This property holds as the Axis enum object to decide the direction used in the main axis. For Row and Column, it is fixed.

- **mainAxisAlignment:** This property takes in MainAxisAlignment enum as the object to decide how the children widgets should be place in mainAxisAlignment. For Row it is horizontal and for Column it is vertical.
- **mainAxisSize:** This property decides the size of main-axis by taking in MainAxisSize enum as the object.
- **runtimeType:** This property tells the run-time type of the Row or Column widget.

- **textBaseline:** This property is responsible for the alignment of the text in the Row or Column widget with respect to a baseline.
- **textDirection:** This property controls the text direction of the *Row* or *Column* widget, which can either be from left-to-right (by default) or right-to-left.
- **verticalDirection:** This property takes in *VerticalDirection enum* as the object to determine the order in which the children should be layered.

<https://github.com/TopsCode/Flutter/tree/main/Module%203/4.%20StatefulWidget>

<https://github.com/TopsCode/Flutter/tree/main/Module%203/5.%20StatelessWidget>

<https://github.com/TopsCode/Flutter/tree/main/Module%203/6.%20Row%20%26%20Column%20Widget>

Flutter Icon

An icon is a graphic image representing an application or any specific entity containing meaning for the user. It can be selectable and non-selectable. For example, the company's logo is non-selectable. Sometimes it also contains a hyperlink to go to another page. It also acts as a sign in place of a detailed explanation of the actual entity.

Other UI Controls

- AlertDialog
- Radiobutton
- checkbox
- switch
- TextFormField
- Style Properties

<https://github.com/TopsCode/Flutter/tree/main/Module%203/7.%20Flutter%20Icon>

<https://github.com/TopsCode/Flutter/tree/main/Module%203/8.%20Other%20Ui%20Components>

Working with Form

Flutter provides a Form widget to create a form. The form widget acts as a container, which allows us to group and validate the multiple form fields. When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields.

The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Form Validations

Validation is a method, which allows us to correct or confirms a certain standard. It ensures the authentication of the entered data.

Validating forms is a common practice in all digital interactions. To validate a form in a flutter, we need to implement mainly three steps.

Step 1: Use the Form widget with a global key.

Step 2: Use TextFormField to give the input field with validator property.

Step 3: Create a button to validate form fields and display validation errors.

Let us understand it with the following example. In the above code, we have to use `validator()` function in the `TextField` to validate the input properties. If the user gives the wrong input, the `validator` function returns a string that contains an error message; otherwise, the `validator` function return null. In the `validator` function, make sure that the `TextField` is not empty. Otherwise, it returns an error message.

- Here We are Creating Register and Login Form as well as Custom Forms

<https://github.com/TopsCode/Flutter/tree/main/Module%203/9.%20Working>

Flutter Images

In this section, we are going to see how we can display images in Flutter. When you create an app in Flutter, it includes both code and assets (resources). An asset is a file, which is bundled and deployed with the app and is accessible at runtime. The asset can include static data, configuration files, icons, and images. The Flutter supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP.

Displaying images is the fundamental concept of most of the mobile apps. Flutter has an `Image` widget that allows displaying different types of images in the mobile application.

Flutter Assets Images

Flutter is an open-source, cross-platform UI development kit developed by Google. It is gaining popularity these days, as the app made in flutter can run on various devices regardless of their platform. It is majorly used to develop applications for Android and iOS, as a single app made in flutter can work efficiently on both

platforms. Here we will learn how to add images in the flutter app. A flutter app when built has both assets (resources) and code. Assets are available and deployed during runtime. The asset is a file that can include static data, configuration files, icons, and images. The Flutter app supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and

<https://github.com/TopsCode/Flutter/tree/main/Module%203/11.%20Flutter%20images>

Floating Action Button

A floating action button is a circular icon button that hovers over content to promote a primary action in the application. Floating action buttons are most commonly used in the [Scaffold.floatingActionButton](#) field

<https://github.com/TopsCode/Flutter/tree/main/Module%203/12.%20Floating%20ActionButton>

Pageview in Flutter

A PageView widget allows the user to swipe/transition between different screens in your app. All you need to set it up are a PageViewController and a PageView.

You can use PageController to control which page is visible in the view. In addition to being able to control the pixel offset of the content inside the PageView, PageController lets you control the offset in terms of pages, which are increments of the viewport size.



<https://github.com/TopsCode/Flutter/tree/main/Module%203/13.%20Pageview%20in%20Flutter/PageView-Flutter-main/PageView-Flutter-main>

Module -4

Advance UI Designing and Development

InitState Method

There are two types of widgets provided in Flutter.

- 1. The Stateless Widget

- 2. The Stateful Widget

As the name suggests Stateful Widgets are made up of some ‘States’.

The initState() is a method that is called when an object for your stateful widget is created and inserted inside the widget tree.

It is basically the entry point for the Stateful Widgets. `initState()` method is called only and only once and is used generally for initializing the previously defined variables of the stateful widget. `initState()` method is overridden mostly because as mentioned earlier it is called only once in its lifetime. If you want to trigger it again you have to shift the control to an entirely new screen and a new state.

<https://github.com/TopsCode/Flutter/tree/main/Module%204/Init%20State>

Working with Navigation and Routing

Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information. For example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product.

In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity, whereas, in iOS, it is equivalent to a ViewController.

In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.

You can also perform Navigation with routing name method

Navigator.pushNamed()

https://github.com/TopsCode/Flutter/tree/main/Module%204/Navigation%20and%20_Routing

Pass Data to another screen

Often, you not only want to navigate to a new screen, but also pass data to the screen as well. For example, you might want to pass information about the item that's been tapped.

Remember: Screens are just widgets. In this example, create a list of todos. When a todo is tapped, navigate to a new screen (widget) that displays information about the todo. This recipe uses the following steps:

1. Define a todo class.
2. Display a list of todos.

3. Create a detail screen that can display information about a todo.
4. Navigate and pass data to the detail screen.

```
Navigator.push(context, MaterialPageRoute(builder: (context) =>  
DetailScreen(todo: todos[index]),),
```

<https://github.com/TopsCode/Flutter/tree/main/Module%204/Pass%20Data%20to%20another%20screen>

Listview

In Flutter, ListView is a scrollable list of widgets arranged linearly. It displays its children one after another in the scroll direction i.e, vertical

horizontal
There are different types of ListViews

- **ListView**
- **ListView.builder**
- **ListView.separated**
- **ListView.custom**

- List View is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.
- For example when we need to show items in a vertical scrolling list. One interesting aspect is this component can be deeply customized.
- <https://github.com/TopsCode/Flutter/tree/main/Module%204/Listview>
and can be adapted to our
<https://github.com/TopsCode/Flutter/tree/main/Module%204/Gridview>
needs.

GridView



Bottom Navigation bar

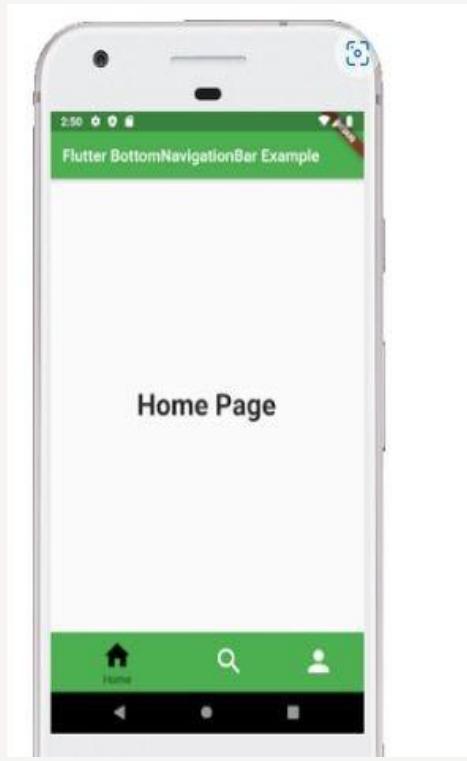
The Bottom Navigation bar has become popular in the last few years for navigation between different UI. Many developers use bottom navigation because most of the app is available now using this widget for navigation between different screens.

The bottom navigation bar in Flutter **can contain multiple items** such as text labels, icons, or both. It allows the user to navigate between the top-level views of an app quickly. If we are using a larger screen, it is better to use a **side navigation bar**.

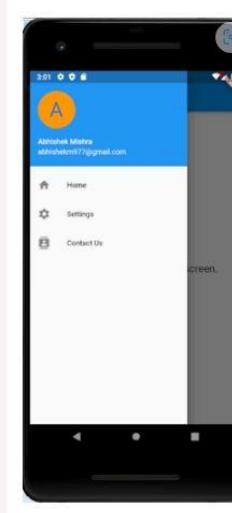
In Flutter application, we usually set the bottom navigation bar in conjunction with the scaffold widget. Scaffold widget provides a Scaffold.bottomNavigationBar argument to set the bottom navigation bar. It is to note that only adding BottomNavigationBar will not display the navigation items. It is required to set the BottomNavigationItems for

Items property that accepts a list of BottomNavigationItems widgets

<https://github.com/TopsCode/Flutter/tree/main/Module%204/bottom%20navigation%20bar>



Navigation Drawer



<https://github.com/TopsCode/Flutter/tree/main/Module%204/navigation%20drawer>

Splash Screen

Splash Screen is the first screen that we see when we run application. It is also known as **Launch Screen**. We will implement three basic methods to add a splash screen in our app.

<https://github.com/TopsCode/Flutter/tree/main/Module%204/Splash%20Screen>

Flutter Gestures

Gestures are used to interact with an application. It is generally used in touch-based devices to physically interact with the application. It can be as simple as a single tap on the screen to a more complex physical interaction like swiping in a specific direction to scrolling down an application. It is heavily used in gaming and more or less every application requires it to function as devices turn more

touch-based than ever. In this article, we will discuss them in detail. Some widely used gestures are mentioned here :

- **Tap:** Touching the surface of the device with the fingertip for a small duration time period and finally releasing the fingertip.
- **Double Tap:** Tapping twice in a short time
- **Drag:** Touching the surface of the device with the fingertip and then moving fingertip in a steadily and finally releasing the fingertip.
- **Flick:** Similar to dragging, but doing it in a speedier way
- **Pinch:** Pinching the surface of the device using two fingers.
- **Zoom:** Opposite of pinching.

- **Panning:** Touching the device surface with the fingertip and moving it in the desired direction without releasing the fingertip

The GestureDetector widget in flutter is used to detect physical interaction with the application on the UI. If a widget is supposed to experience a gesture, it is kept inside the GestureDetector widget. The same widget catches the gesture and returns the appropriate action or response.

- Tap
- Double
- Long Press
- Vertical Drag
- Horizontal Drag
- Pan

Module 5

Flutter -Offline Database

Sqflite

A database is an organized collection of data, which supports the storage and manipulation of data and accessed electronically from a computer system. We can organize data into rows, columns, tables, and indexes. It makes data management easy. We can store many things in the database, like name, age, picture, image, file, pdf, etc.

Flutter provides many packages to work with the database. The most used and popular packages are:

- **SQLite database:** It allows to access and manipulate SQLite database.
- **Firebase database:** It will enable you to access and manipulate the cloud database.

SQLite Database

SQLite is a popular database software library that provides a relational database management system for local/client storage.

It is a light-weight and time-tested database engine and contains features like self-contained, server-less, zero-configuration, transactional SQL database engine.

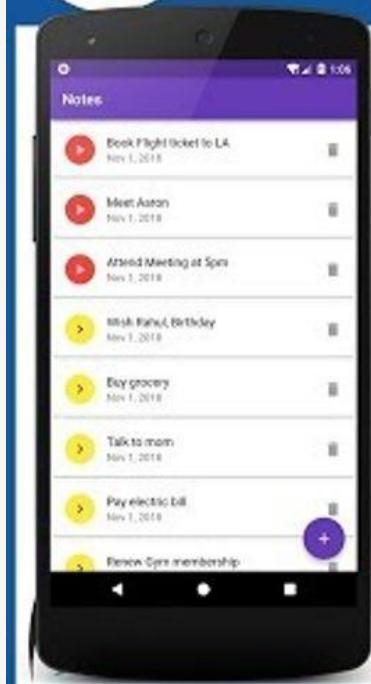
Flutter SDK does not support SQLite directly. Instead, it provides a plugin **sqflite**, which performs all operations on the database as similar to the SQLite library. The sqflite provides most of the core functionalities related to the database are as follows:

- It creates or opens the SQLite database.
- It can execute SQL statements
- ~~easily~~ It also provides an advanced query method to get information from the SQLite database.
<https://github.com/TopsCode/Flutter/tree/main/Module%205/sql>
- <https://github.com/TopsCode/Flutter/tree/main/Module%205/sharedprefenc>

SQFLite Database

NoteKeeper App

← Building UI →



Module - 6

Advance App Development

Gallery and Camera access

We can add images from the gallery using the image_picker package in Flutter. For this, you'll need to use your real device.

Follow the below steps to display the images from the gallery:

<https://github.com/TopsCode/Flutter/tree/main/Module%206/Gallery%20and%20Camera%20Access%20in%20Flutter>



Fetch Data from Internet

Fetching data from the internet is necessary for most apps.

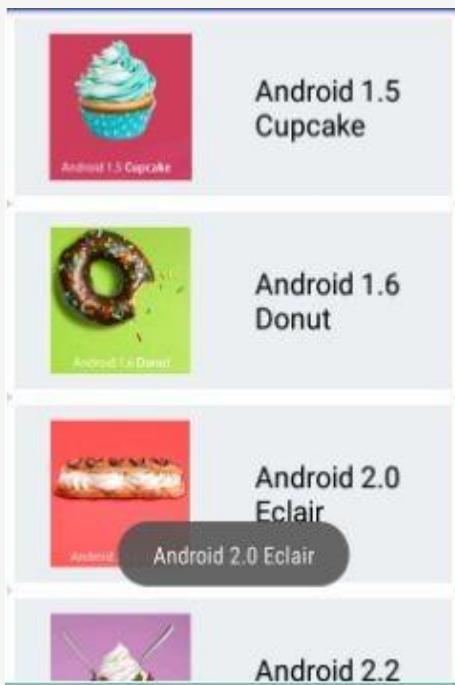
Dart and Flutter provide tools, such as the `http` package, for this type of work.

This recipe uses the following steps:

1. Add the `http` package.
2. Make a network request using the `http` package.
3. Convert the response into a custom Dart object.
4. Fetch and display the data with Flutter.

1. Add the http package
2. Make a network request
3. Convert the response into a custom Dart object
4. Convert the http.Response to an Album
5. Fetch the data
6. Display the data

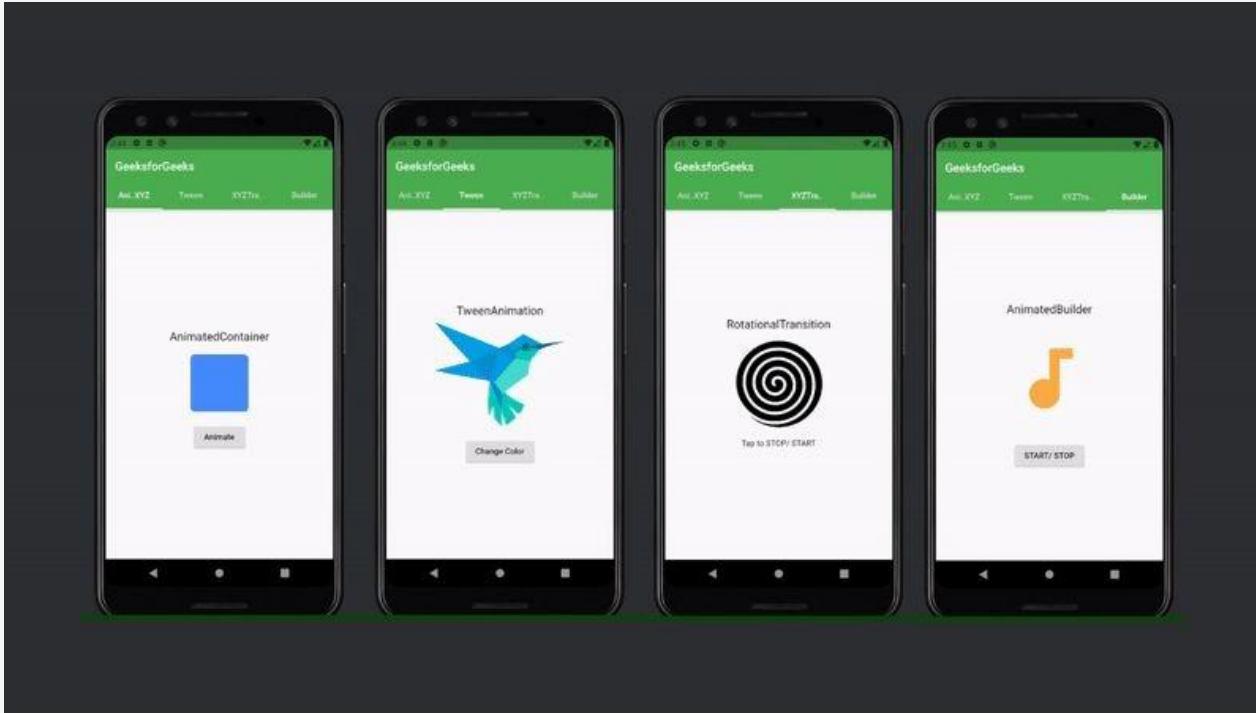
<https://github.com/TopsCode/Flutter/tree/main/Module%206/JSON%20CRUD-with%20API>



Animation

Types of Animation

1. SizeTransition
 2. FadeTransition
 3. AlignTransition
 4. RotationTransition
 5. PositionedTransition
 6. DecoratedBoxTransition
- n



There is one more latest animation available which is used in trending that animation is Lottie Animation .

- If you want to implement Lottie Animation then go to <https://lottiefiles.com/>
- Select Any Animation Download the JSON File
- Add the Dependency for Lottie Animation and implement it
<https://github.com/TopsCode/Flutter/tree/main/Module%206/Animation>

Working with Firebase

Firebase helps developers to manage their mobile app easily. It is a service provided by Google. Firebase has various functionalities available to help developers manage and grow their mobile apps. In this article, we will learn how to write and read data into/from firebase. We will be using flutter for this. To do this job we need to follow the 3-step procedure:

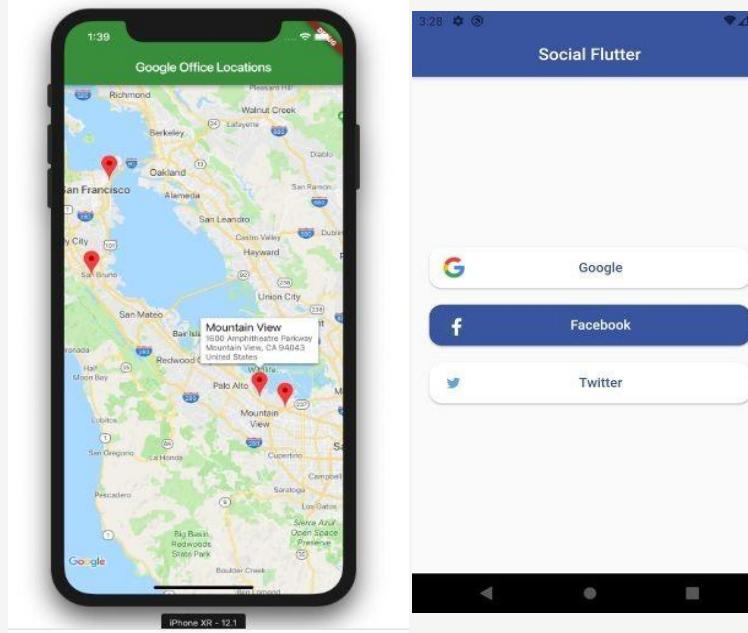
1. Adding firebase to our app
2. Firebase Setup
3. Implement using Code

Notification with Firebase

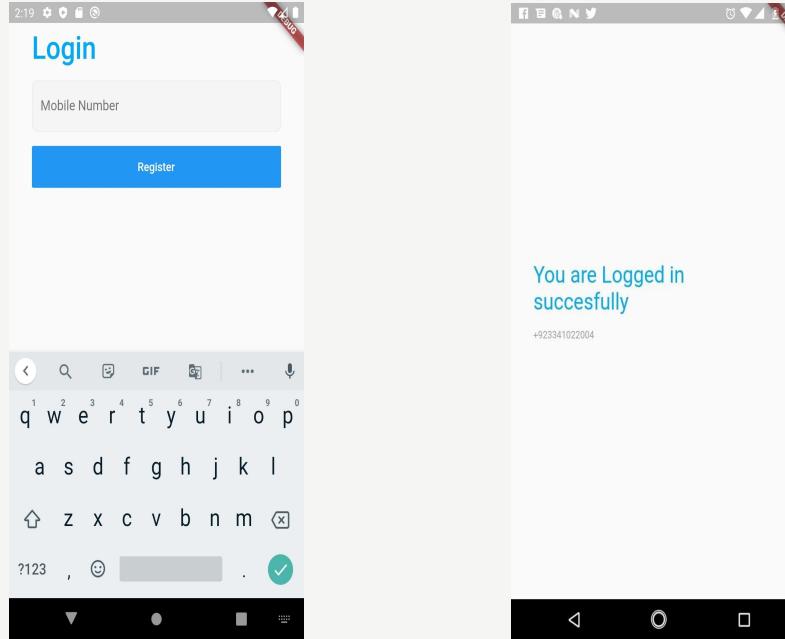
<https://github.com/TopsCode/Flutter/tree/main/Module%206/Firebase%20Crud>

[https://github.com/TopsCode/Flutter/tree/main/Module%206/Firebase%20Notification/
flutter-FCM-main](https://github.com/TopsCode/Flutter/tree/main/Module%206/Firebase%20Notification/flutter-FCM-main)

Google Map and Social media integration



OTP verification



https://github.com/TopsCode/Flutter/tree/main/Module%206/Google%20Map/flutter_google_maps_example-master

<https://github.com/TopsCode/Flutter/tree/main/Module%206/OTP%20verification/Flutter-OTP-Authentication>

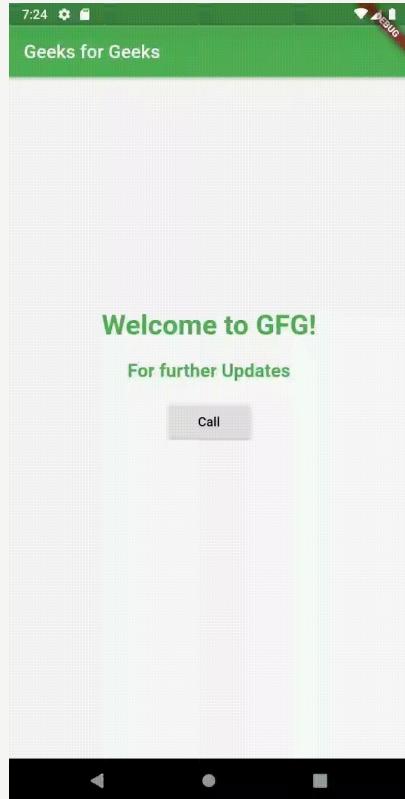
<https://github.com/TopsCode/Flutter/tree/main/Module%206/Background%20Local%20Notification>

<https://github.com/TopsCode/Flutter/tree/main/Module%206/JSON%20Register-Login>

<https://github.com/TopsCode/Flutter/tree/main/Module%206/Shimmer%20Effect>

Making Phone Calls

In this online world, customer care is playing a major role in the success of a company. Users are quite satisfied when they converse with the executives through calls. This has forced companies to add phone numbers to their apps for their customers to contact them easily. But, dialing those numbers from the app into the default phone app makes it quite cumbersome. So, in order to improve the user experience, Flutter has come up with a feature where the user can call the other, just with a click. This can be achieved by using the “url_launcher” plugin.



Sms Launcher and Email Launcher



Welcome to GFG!

For any Queries, Mail us

Here

Or Send SMS

Here

<https://github.com/TopsCode/Flutter/tree/main/Module%206/Call>

Module 7

Flutter Deployment

For Android Device

When a new Flutter app is created, it has a default launcher icon. To customize this icon, you might want to check out the [flutter_launcher_icons](#) package.

Alternatively, you can do it manually using the following steps:

1. Review the [Material Design product icons](#) guidelines for icon design.
2. In the [project]/android/app/src/main/res/ directory, place your icon files in folders named using [qualifiers](#). The default mipmap- folders demonstrate the correct naming convention.

3. In AndroidManifest.xml, update the application tag's android:icon attribute to reference icons from the previous step (for example, <application android:icon="@mipmap/ic_launcher" ...).
4. To verify that the icon has been replaced, run your app and inspect the app icon in the **Enabling Material Components**
6. Signing the app
7. Create an upload keystore

For windows:

```
keytool -genkey -v -keystore  
c:\Users\USER_NAME\upload-keystore.jks -storetype  
JKS -keyalg RSA -keysize 2048 -validity 10000  
-alias upload
```

8. Configure signing in

gradle

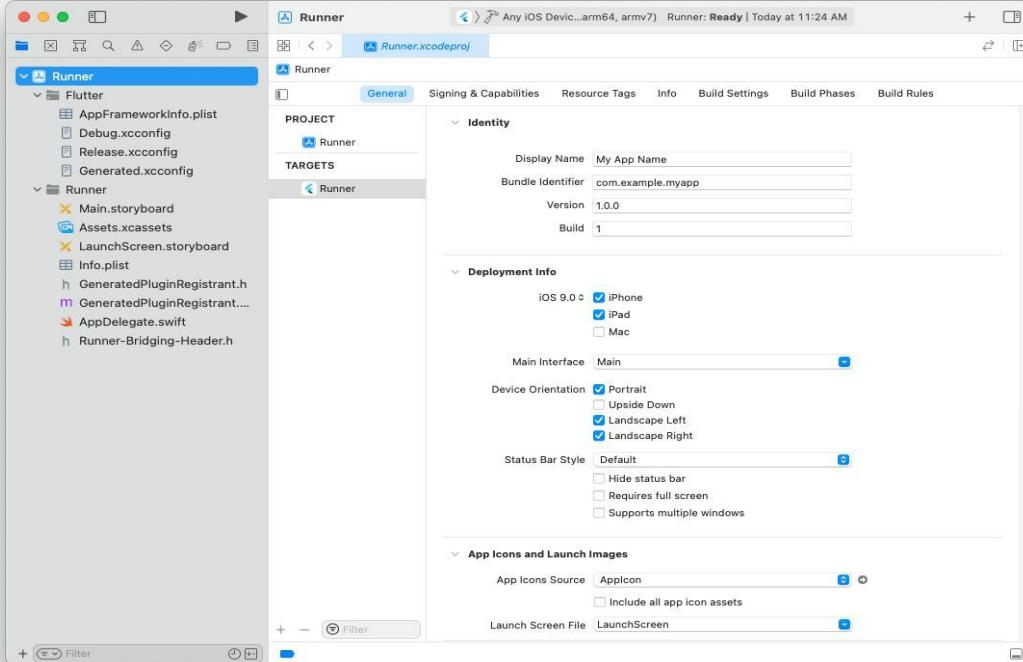
9. Enabling multidex support

10. Create an release App From Generated Signed APK

For IOS Device

Xcode is required to build and release your app. You must use a device running macOS to follow this guide.

- Register your app on App Store Connect
- Register a Bundle ID
- Create an application record on App Store Connect
- Review Xcode project settings



- Updating the app's deployment
- ~~Action~~ app icon
- Create a build archive and upload to App Store Connect
- Update the app's build and version numbers
- Upload the app bundle to App Store Connect
- For CLI tools follow Given Link

<https://docs.flutter.dev/deployment/ios>

Thank You