This doc is created to make note of all the topics that we are going to teach in noesis2019.

Feel free to make any edits in the doc. The list will update in real time as you make edits so make sure that you edit it wisely. (ADD THE TOPICS WHICH WE ARE GOING TO TEACH.)

**OOPs**

1.Structures

1.a. Referencing structure elements.

1.b. Nested Structures

1.c. Array of structures

1.d. Array within structures

1.e. Passing structures to functions

2. OOPS Introduction

2.a. What is OOPS

2.b. Various programming Paradigms

2.b.1 Procedral Oriented Programming

2.b.2 Object Based Programming

2.b.3 Object Oriented Programming

2.c Basic OOPS conepts

2.c.1 Data Abstraction

2.c.2 Encapsulation

2.c.3 Modularity

2.c.4 Inheritance

2.c.5 Polymorphism

3. Classes and objects

3.a. Declaration of a class and ways of defining methods

3.b. Referrencing class members

3.c. Array within a class

3.d. Scope of class and its class members

3.e. Types of class functions

3.f. Nested class

3.g. Access specifiers in a class

3.h. Data hiding and encapsulation and abstrsaction

3.i. Function in a class

3.i.1 Inline functions and constant member functions

3.i.2 Nesting of member functions

3.i.3 Scope resolution operator

3.j. Using objects of a class

3.j.1 Memory allocation of objects

3.j.2 Array of objects

3.j.3 Objects as function arguments

3.j.4 Returning objects

3.k. Constant objects and constant member functions

3.l. Static members and member functions

3.m Composition ‘HAS A’ and ‘IS A’ relationship

4.n Global , local and static local objects

4. Function Overloading

4.a Need

4.b Declaration and definition

4.c Limitations

4.d Default arguments vs Function Overloading

4.e Steps involved in finding a best match

5. Constructors and destructors

5.a Introduction

5.b Need of constructors and destructors

5.c Declaration and definition

5.d Default constructors

5.e Parameterized constructors

5.f Significance of default constructors

5.g Copy constructor

5.h Dynamic initialisation of objects

5.i Constructor overloading and advantages

5.j Order of constructor and destructor invocation in a nested class

6. Inheritance

6.a Need for inheritance

6.b Types of inheritance

6.b.1 Single inheritance

6.b.2 Multiple inheritance

6.b.3 Multilevel inheritance

6.b.4 Hybrid inheritance

6.c Visibility modes

6.d Inheritance and access control

6.d.1 Access control in publicly , privately and protectedly derived classes

6.d.2 Making private members as public

6.d.3 Friend function and friend classes

6.d.4 Shadowing or Overriding

6.e Order of invocation of constructors and destructors in multiple inheritance

7. Pointers with Structures and Class

7.a Pointers arithmatic

7.b Memory leaks

7.c Pointers , Structures , Class

7.c.1 Declaration and Definition of pointers to structure and class

7.c.2 Pointers within a structure and class

7.c.3 Self referential structures

7.c.4 Dynamic structures - new and delete operator

7.d Objects as function arguments

7.d.1 Passing objects as call by value

7.d.2 Passing objects as call by reference

7.d.3 return by value and return by reference

7.e Pointers and objects

7.e.1 Declaration

7.e.2 Use of object pointers

7.e.3 This pointer

7.e.4 Cascaded member function call

8. OOPs implementation

8.a Ambiguity in inheritance and use if scope resolution operator

8.b Polymorphism

8.b.1 How to use base class pointer to access derived class object

8.b.2 Ambiguity in above method and compile time binding

8.b.3 Runtime binding

8.b.4 Virtual methods and pointers

8.b.5 Dynamically created objects and virtual destructors

8.b.6 Virtual base class

8.b.7 Pure virtual

8.b.8 Abstract and base classes

8.c Operator overloading

8.c.1 Unary , BInary and Assignment

8.c.2 Declaration and definiton

8.d Templates

8.d.1 Function templates - declaration and definition

8.d.2 When templates are used and when function overloading is used

8.d.3 Class templates

**STL**

1.

2.

3.

4.

5.

**Git/Github**