**INDEX**

[Why OOPs is required 3](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203680)

[Reason for development of OOPs? 4](#_Toc171203681)

[What is OOPs & Why it is Required? 4](#_Toc171203682)

[What is POPs (Procedure Oriented Programming)? 5](#_Toc171203683)

[Why C++ is more secure than C? 5](#_Toc171203684)

[Principle of OOP 6](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203685)

[What is principle of OOPs? 7](#_Toc171203686)

[Feature of OOPs? 7](#_Toc171203687)

[Benefits of OOPs? 7](#_Toc171203688)

[Beginning with C++ 8](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203689)

[History of C++? 9](#_Toc171203690)

[Applications of C++? 9](#_Toc171203691)

[Boilerplate of C++? 9](#_Toc171203692)

[Comment in C++? 9](#_Toc171203693)

[I/O Statement? 9](#_Toc171203694)

[Classes & Object 10](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203695)

[What is class? 11](#_Toc171203696)

[What is object? 11](#_Toc171203697)

[What is Public, Private, Protected? 11](#_Toc171203698)

[What is Data Member & Member Function? 11](#_Toc171203699)

[Functions 12](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203700)

[Constructors & Destructors 14](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203701)

[Operator Overloading 16](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203702)

[Type Conversions 18](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203703)

[Inheritance 20](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203704)

[Pointers 22](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203705)

[Virtual Function 24](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203706)

[Polymorphism 26](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203707)

[Managing Console I/O Operation 28](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203708)

[Working with Files 30](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203709)

[Template 32](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203710)

[Exception Handling 34](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203711)

[Intro to the standard Template Library 37](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203712)

[Manipulating String 39](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203713)

[New Feature of ANSI C++ 41](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203714)

[Object Oriented System Development 43](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C++\C++.docx#_Toc171203715)

**Tokens, Expressions & Control Structures**

**1**

Why OOPs is required

**Objectives**

Reason for development of OOPs?

Rapid Advancement of now days has created many issues which are needed to be addressed:

1. How to represent real-life entities of problems in system design?
2. How to design systems with open interfaces?
3. How to ensure reusability and extensibility of modules?
4. How to develop modules that are tolerant to any changes in future?
5. How to improve software productivity and decrease software cost?
6. How to improve the quality of software?
7. How to manage time schedules?
8. How to industrialize the software development process?

Now days software require below features:

1. Correctness
2. Maintainability
3. Reusability
4. Openness & Interoperability
5. Portability
6. Security
7. Integrity
8. User Friendliness

What is OOPs & Why it is Required?

Object oriented programming is a paradigm (Pattern) that is widely used in software development. Moreover, it uses objects that can represent abstract (entities that have no physical constraints as they not have direct representation in physical world) concepts or real-world objects to represent and manipulate data, including the actions that can be performed.

OOPs has some concept which allow us to work with real world entities:

1. Object
2. Classes
3. Data Abstraction & Encapsulation
4. Inheritance
5. Polymorphism
6. Dynamic Binding
7. Message Passing

Machine Language

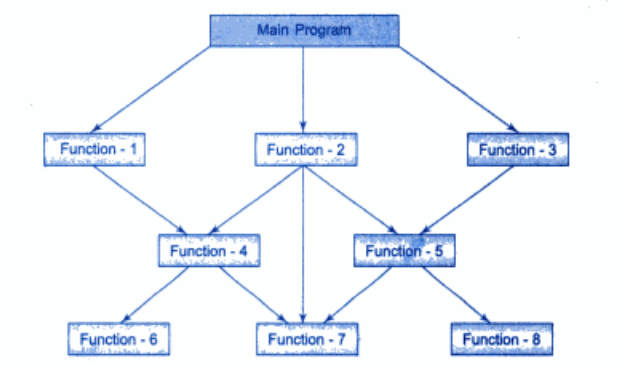
Assembly Language

Procedure Oriented

OOPs

**0 , 1**

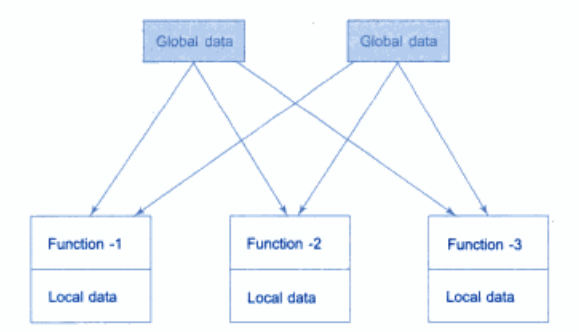
What is POPs (Procedure Oriented Programming)?

In POPs, the problem is viewed as sequence of things to be done such as reading, calculating & printing then a number of functions are written to accomplish these tasks. The primary focus is on functions.

Some characteristics exhibited by procedure-oriented programming are:

1. Emphasis is on doing things (algorithms).
2. Large programs are divided into smaller programs known as functions.
3. Most of the functions share global data.
4. Data move openly around the system from function to function.
5. Functions transform data from one form to another.
6. Employs top-down approach in program design.

Why C++ is more secure than C?

As C is POPs based language so many functions are created to do the work but during this many important data is placed as global so it can be access by all the functions and due to this, we need to revise all the function even if we want to revise an external data structure and also results in many bugs and error and value of variables can be changed by other function also.

Where as in C++ we can put different type of restriction on it by which we can decide the scope of data to be accessed.

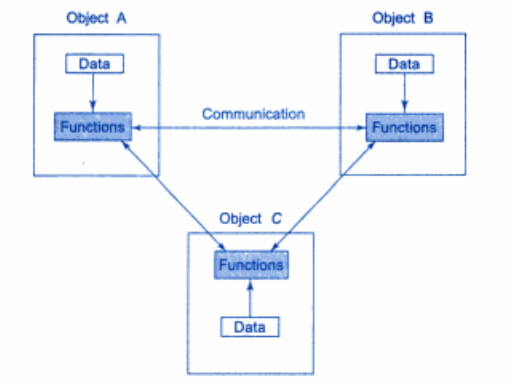
**2**

Principle of OOP

**Objectives**

What is principle of OOPs?

Principle of OOPs is that it treats data as critical element in the program development & does allow it to freely flow around in the system. It breaks problem into small entities called object and then build data & function around these objects.

The data of object is only access by those function which are associated with that object.

Feature of OOPs?

1. Emphasis is on data rather than procedure.
2. Programs are divided into what are known as objects.
3. Data structures are designed such that they characterize the objects.
4. Functions that operate on the data of an object are tied together in the data structure.
5. Data is hidden and cannot be accessed by external functions.
6. Objects may communicate with each other through functions.
7. New data and functions can be easily added whenever necessary.
8. Follows bottom-up approach in program design.

Benefits of OOPs?

1. Through inheritance, we can eliminate redundant code and extend the use of existing classes.
2. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.
3. The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
4. It is possible to have multiple instances of an object to co-exist without any interference.
5. It is possible to map objects in the problem domain to those in the program.
6. It is easy to partition the work in a project based on objects.
7. The data-centered design approach enables us to capture more details of a model in implementable form.
8. Object-oriented systems can be easily upgraded from small to large systems.
9. Message passing techniques for communication between objects makes the interface descriptions with external systems much simpler.
10. Software complexity can be easily managed.

**3**

Beginning with C++

**Objectives**

History of C++?

C++ is an object-oriented programming language. It was developed by Bjarne Stroustrup at AT&T Bell Laboratories in Murray Hill, New Jersey, USA, in the early 1980's. Stroustrup, an admirer of Simula67 and a strong supporter of C, wanted to combine the best of both the languages and create a more powerful language that could support object-oriented programming features and still retain the power and elegance of C. The result was C++.

Old name of C++ is “C with classes”. Later in 1983 it is given the name C++ by ASCII.

In Nov 1997, the ANSI/ISO standards committee standardised these changes & added several new features.

It has all those feature C has but it all has additional features and some feature has different syntax than C.

Applications of C++?

1. Since C++ allows us to create hierarchy-related objects, we can build special object-oriented libraries which can be used later by many programmers.
2. While C++ is able to map the real-world problem properly, the C part of C++ gives the language the ability to get close to the machine-level details.
3. C++ programs are easily maintainable and expandable. When a new feature needs to be implemented, it is very easy to add to the existing structure of an object.
4. It is expected that C++ will replace C as a general-purpose language in the near future.

Boilerplate of C++?

**#include<iostream.h>    // include header file**

**int main{**

**cout>>"Hello Vanshit";**

**return 0;**

**}                       // End of code**

**Output: Hello Vanshit**

**Note :** Execution of code begins at the **main()** in all programs.

Comment in C++?

// 🡪 Single line Comment

/\* \*/ 🡪 Multi line Comment

Double line comment can’t be used in below manner

for(int i=1; i<=10 **/\* loops run 10 times \*/**; i++)

as everything after double slash (//) become comment so it can’t be use in between a code like /\* \*/.

I/O Statement?

cout 🡪 It is used in place of printf. It stands for console output.

cin 🡪 It is use in place of scanf. It stands for console input.

**4**

Classes & Object

**Objectives**

What is class?

1. It is very similar to “Structure” of C.
2. It is like a blueprint by which we can create objects.
3. It is user-define data type which contains data member & member function.
   1. Data member 🡪 It means variable declared in class.
   2. Member function 🡪 It means function declared in class.
4. Data is more secure in class as compare to function in C, as class has 3 type of protection 🡪 public, private, protected which allow us to decide to scope in which data can be accessed.
5. Class do not make any memory allocation until & unless an object is created from it. Once the object is created then that object will do memory allocation according to code it will access.

What is object?

What is Public, Private, Protected?

What is Data Member & Member Function?

**5**

Functions

**Objectives**

**6**

Constructors & Destructors

**Objectives**

**7**

Operator Overloading

**Objectives**

**8**

Type Conversions

**Objectives**

**9**

Inheritance

**Objectives**

**10**

Pointers

**Objectives**

**11**

Virtual Function

**Objectives**

**12**

Polymorphism

**Objectives**

**13**

Managing Console I/O Operation

**Objectives**

**14**

Working with Files

**Objectives**

**15**

Template

**Objectives**

**16**

Exception Handling

**Objectives**

**17**

Intro to the standard Template Library

**Objectives**

**18**

Manipulating String

**Objectives**

**19**

New Feature of ANSI C++

**Objectives**

**20**

Object Oriented System Development

**Objectives**