1. object oriented Programming

→ Programming paradigm where everything is represented as an object.

OOPs (object oriented programming system)

→ object → Real world entity like pen, chair, table. OOP is a methodology to design a program using objects and classes. Makes code more manageable.

Important concepts → object, class, Inheritance, Polymorphism, Abstraction, Encapsulation.

→ class → collection of objects is called class. It is a logical entity.

→ Inheritance → when one object acquires all the properties and behaviours of parent object; it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

→ Polymorphism → when one task is performed by different ways, it is known as polymorphism.

For example, to convince the customer differently, to draw something eg shape or rectangle etc.

In C++, function overloading and function overriding are used to achieve polymorphism.

→ Abstraction → Hiding internal details and showing functionality is known as abstraction.

For ex, in phone call, we don't know internal processing.

In C++, we use abstract class and interface to achieve abstraction.

→ Encapsulation → Binding (or wrapping) code and data together in a single unit is known as encapsulation.

→ Advantages of OOPs over procedure oriented programming

(i) OOPs make development and maintenance easier.

(ii) OOPs provide data hiding whereas global data can be accessed from anywhere in procedure oriented programing.

(iii) OOPs simulate real world more effectively.

examples

Class : Fruit object : Apple, Guava, Mango.

Class : Mobile phone Object : Iphone, Samsung, Moto

Class : Food object : Pizza, Burger, Samosa.

⇒ c++ object

(i) object is an entity that has state and behaviour

state means data and behaviour means functionality.

(ii) Object is a runtime entity. It is created at runtime.

(iii) object is an instance of a class. All the members of class can be accessed through the object.

(iv) ex. Student S1; S1 → instance of student class.

Student → class

S1 → object.

⇒ c++ class

(i) class is a group of similar objects. It is a template from which objects are created. It can have fields, methods, constructors etc.

ex.

```
class student {
    public: int id;      // field or data member.
            string name; // field or data member.
};
```

→ example of object and class

```
#include <iostream>
using namespace std;

class student {

    public :
        int id;          // data member (instance variable).

        string name;

};

int main () {

    student s1;          // creating an object of student

    s1. id = 201;

    s1. name = "manan Garg";

    cout << s1. id << "\n";

    cout << s1. name << "\n";

    return 0;

}
```

Output → 201
manan garg.

example → Initialise and display data through member.

```
class student {
    public:
        int id;
        string name;
        void insert ( int i, string s) {
                id = i;
                name = s;
            }
        void display () {
                cout << id << " " << name;
            }
        };
int main () {
    student s1;
    s1. insert ( 200, "Manan");
    s1. display ();
    return 0;
}.
```

Output

200 Manan.

⇒ C++ Constructor.

(i) Constructor is a special method which is invoked automatically at the time of object creation.

(ii) It is used to initialize the data members of new object generally.

(iii) Constructor in C++ has same name as class or structure

(iv) There can be two types of constructors

(a) Default constructor.

(b) Parametrised constructor.

→ C++ Default constructor.

(i) A constructor which has no argument is called as default constructor.

(ii) It is invoked at the time of creating object.

eg

```
class employee {
    public :
        employee () {
            cout << "default constructor
                invoked" << "\n" ;
        }
int main () {
        employee e1;
    3   return 0;
```

Output

default constructor invoked.

⇒ C++ Parametrized constructor.

A constructor which has parameters is called parametrised constructor. It is used to provide different values to distinct objects.

ex.

```
class employee {
    public:
        int id;
        string name;
        float salary;

        employee (int i, string s, float salary) {
            id = i;
            name = s;
            salary = salary;
        }

        void display () {
            cout << id << " " << name << " " <<
                                    salary;
        }.

int main() {
    employee e1 = employee (100, "Manan", 10000);
    e1.display()
    }.
```

Output

100 Manon 10000.