Function Overloading

Whenever a program contains more than one function with same name but different types of parameters, then it is known as function overloading.

Syntax:-

```
class class_name {
   public:
   void disp() {
   //operations
   }
   void disp(int a) {
   //opeartions
   }
}
```

Function Overriding

Whenever we declare a function in base class and derive class in such way that function name and parameters must be same called function overriding.

Syntax:-

```
class class_base {
   public:
   void disp() {
        //operations
   }
class class_derive: public class_base {
   void disp() {
       //opeartions
   }
}
```

Difference between function overloading and function overriding

Function Overloading	Function Overriding
Function Overloading provides multiple definitions of the function by changing signature.	Function Overriding is the redefinition of base class function in its derived class with same signature.
An example of compile time polymorphism.	An example of run time polymorphism.
Function signatures should be different.	Function signatures should be the same.
Overloaded functions are in same scope.	Overridden functions are in different scopes.
Overloading is used when the same function has to behave differently depending upon parameters passed to them.	Overriding is needed when derived class function has to do some different job than the base class function.
A function has the ability to load multiple times.	A function can be overridden only a single time.
In function overloading, we don't need inheritance.	In function overriding, we need an inheritance concept.

Abstraction

It is the process of simplifying complex objects by focusing on essential features while hiding unnecessary details.

Abstract class

A class which contain at least one pure virtual function is known as abstract class, we cannot declare the object of abstract class.

Syntax:-

```
class Mario {
    public:
        virtual void disp() = 0;
}
```

Pure Virtual function

- Those virtual functions which have no definition, they start with virtual keyword and ends with equal to zero.
- If we don't override the pure virtual function in derive class, then derive class also becomes abstract class.
- We cannot change the signature of pure virtual function.

Inline function

An inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of the inline function call. If a function is inline, then the compiler laces the copy of the function's code in the place of the function call. And this can speed up the program execution.

Syntax:-

```
inline return-type function_name(parameters) {
    //operations
}
```

Friend function

It has been declared as a friend of a class not as a member of the class, instead of that it can access private and protected member of class.

Syntax:-

```
friend return-type function_name(class ref);
```

Friend class

A class that granted accessibility of private and protected member of another class, is called friend function.

Syntax:-

```
class class_name {
      private:
         //members
      public:
         friend class class_name;
}
```

Templates

Template is the frame which define its actual meaning in a c++ program. We can draw any logic using template. It will create appropriate code at the time of execution. We can use template by two ways.

- Function template
- Class template

Function template is known as generic function and Class template is also known as generic class.

Function template

```
Template<class_type> //supports any type of datatype
return_type function_name(parameter list) {
        body;
}
```

Class template

```
Template<class_type>
class class_name {
        public:
           type var;
           type function_name(type args)
}
```

Static keyword

- Data member
- Member function

Static data member

- Whenever we declare a data member as a static either inside or outside of a class called static data member.
- There is only one copy of static data member even if there are many class objects.
- It is always initialized with zero because it's default value is zero.
- It is shared memory for all objects of the class.
- It retains it's values.

Static member function

- If we create a member function of a class as a static, then it is known as static member function.
- It accesses only static data members.
- It is also accessible if we don't have any object of a class.