static member functions:
===================
1. A static member function is a member function which is declared with the keyword static inside the class.

2. Syntax:
    ======
    static <ret_type> <fun_name>(arg_list);

   Example:
   =======
    static void showcount();

3. To call a static member function we do not require object. Rather we can call it by simply using classname followed by scope resolution operator.

Example:
=======
Emp::showcount()

4. A static member function has an impt restriction which is that it can only access static data members of the class and not its non static data members.

5. We can declare every member function of the class as static except its constructor and destructor. Because constructor and destructor are called only w.r.t object but a static member function is not at all connected to any object of the class. Thus it is a syntax error in C++ to declare constructor or derstructor as static member function.

Inline Functions
============

```
#inlcude <iostream>
using namespace std;
inline int square(int n)
{
  return n*n;
}
int main()
{
  int a=10,b=20,c;
  c=square(a);
  cout<<"Square of "<<a<<" is "<<c<<endl;
  c=square(b);
  cout<<"Square of "<<b<<" is "<<c<<endl;
  return 0;
}
```

Explicit

catalyst

```
class A
{
public:
    void show()
    {
       ≡
    }
};
```

Implicit Inline

Inline member functions or inline functions in C++ are those functions for which the keyword inline is used while defining them. The general syntax of defining a inline function is:

```
inline <ret_type> <fun_name>(arg)
{
  // body
}
```

Example
======

```
inline int square(int n)
{
  return n*n;
}
```

Advantages:
==========

1. When we decalre a function as inline then the compiler copies the function body and paste's it inplace of the function call.
2. Thus we can say that inline functions are those functions whosie call gets replaced by their body during compilation.

3. Due to this all the overheads related to function call are reduced . For example the compiler is not required to check the protoype or pass the arguments and most importantly the compiler does not has to maintain stack in between function calls.

4. Thus by making a function as inline we increase the compilation/execution speed ans reduce the time taken by the code, there by making it more efficient.

Restrictions
=========

But we must remember 3 impt points before declaring a function as inline:
1. The body of an inline function must be short and small.
2. The body should not contain any complex logic like loops, break ,continue etc.
3. The body must appear before the call in the source code.

If any of the above rules are broken then although no syntax error will arise but the compiler will ignore the keyword inline and treat the function as a normal funciton.