Aim:

To understand the concept of **Inline Functions** with the help of on example.

Theory:

C++ inline function is powerful concept that is commonly used with classes. If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.

Any change to an inline function could require all clients of the function to be recompiled because compiler would need to replace all the code once again otherwise it will continue with old functionality.

To inline a function, place the keyword inline before the function name and define the function before any calls are made to the function. The compiler can ignore the inline qualifier in case defined function is more than a line.

A function definition in a class definition is an inline function definition, even without the use of the inline specifier.

Syntax :

```
inline return_type function_name ( **args ) {
    // function body
}
```

Code :

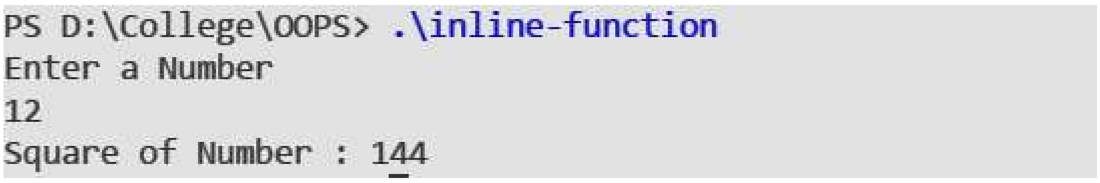
```
#include <iostream>

using namespace std;

inline int square(int x){
    return x*x ;
}

int main (){
    int num ;
    cout << "Enter a Number" << endl ;
    cin >> num ;
    cout << "Square of Number : " << square(num) << endl ;
}
```

Output :



```
PS D:\College\OOPS> .\inline-function
Enter a Number
12
Square of Number : 144
```

Discussion :

The program demonstrates the concept of inline function using the function `square`. The function is expanded in line when it is called. When the function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time. Inline function may increase efficiency if it is small.

Learning Outcomes :

- Function call overhead doesn’t occur.
- It also saves the overhead of push/pop variables on the stack when function is called.
- It also saves overhead of a return call from a function.
- When you inline a function, you may enable compiler to perform context specific optimization on the body of function. Such optimizations are not possible for normal function calls. Other optimizations can be obtained by considering the flows of calling context and the called context.
- Inline function may be useful (if it is small) for embedded systems because inline can yield less code than the function call preamble and return.