 **What Is Inline Function?**

C++ provides an inline functions to reduce the function call overhead. 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 inline function call. This substitution is performed by the C++ compiler at compile time. Inline function may increase efficiency if it is small.

**Video on Inline Functions**:

<http://www.infocobuild.com/education/audio-video-courses/computer-science/ProgrammingInCpp-IIT-Kharagpur/lecture-08.html>

**The syntax for defining the function inline is:**

inline return-type function-name(parameters)

{

   // function code

}

in lining is only a request to the compiler, not a command. Compiler can ignore the request for in-lining. Compiler may not perform in-lining in such circumstances like:

1) If a function contains a loop. (for, while, do-while)

2) If a function contains static variables.

3) If a function is recursive.

4) If a function return type is other than void, and the return statement doesn’t exist in function body.

5) If a function contains switch or goto statement.

Inline functions provide following advantages:

1) Function call overhead doesn’t occur.

2) It also saves the overhead of push/pop variables on the stack when function is called.

3) It also saves overhead of a return call from a function.

4) 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.

5) 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.

**Example 1: The following program demonstrates the use of use of inline function.**

**Output:**

It is also possible to define the inline function inside the class. In fact, all the functions defined inside the class are implicitly inline. Thus, all the restrictions of inline functions are also applied here. If you need to explicitly declare inline function in the class then just declare the function inside the class and define it outside the class using inline keyword.

**Example 2: The following program demonstrates this concept**

**Output:**

**Frequently Asked Questions about Flowchart**

Some of the most frequently asked questions are:

**Q1**: Define Inline Function.

Answer When the function is defined Inline, the C++ compiler puts the function body inside the calling function. You can define function as Inline when the function body is small and need to be called many times, thus reduces the overhead in calling a function like passing values, passing control, returning values, returning control.

**Q2**: Do inline functions improve performance? Explain.

Answer: A function when defined as INLINE, the code from the function definition is directly copied into the code of the calling function.

 It avoids the overhead of calling the actual function. This is because the complier performs and inline expansion which eliminates the time overhead when a function is called.

It Reduces space as no separate set of instructions in memory is written.

**Q3**: When to use Inline function?

Answer: We can use Inline function as per our needs. Some useful recommendation are mentioned below-

We can use the inline function when performance is needed.

We can use the inline function over macros.

We prefer to use the inline keyword outside the class with the function definition to hide implementation details of the function.

**Q4:** What will be the output of the following C code?

#include <stdio.h>

void **inline** func1(int a, int b)

{

   printf ("a=%d and b=%d**\n**", a, b);

}

int **inline** func2(int x)

{

   return x\*x;

}

int main()

{

   int tmp;

   func1(1,4);

   tmp = func2(6);

   printf("square val=%d**\n**", tmp);

   return 0;

}

Output: