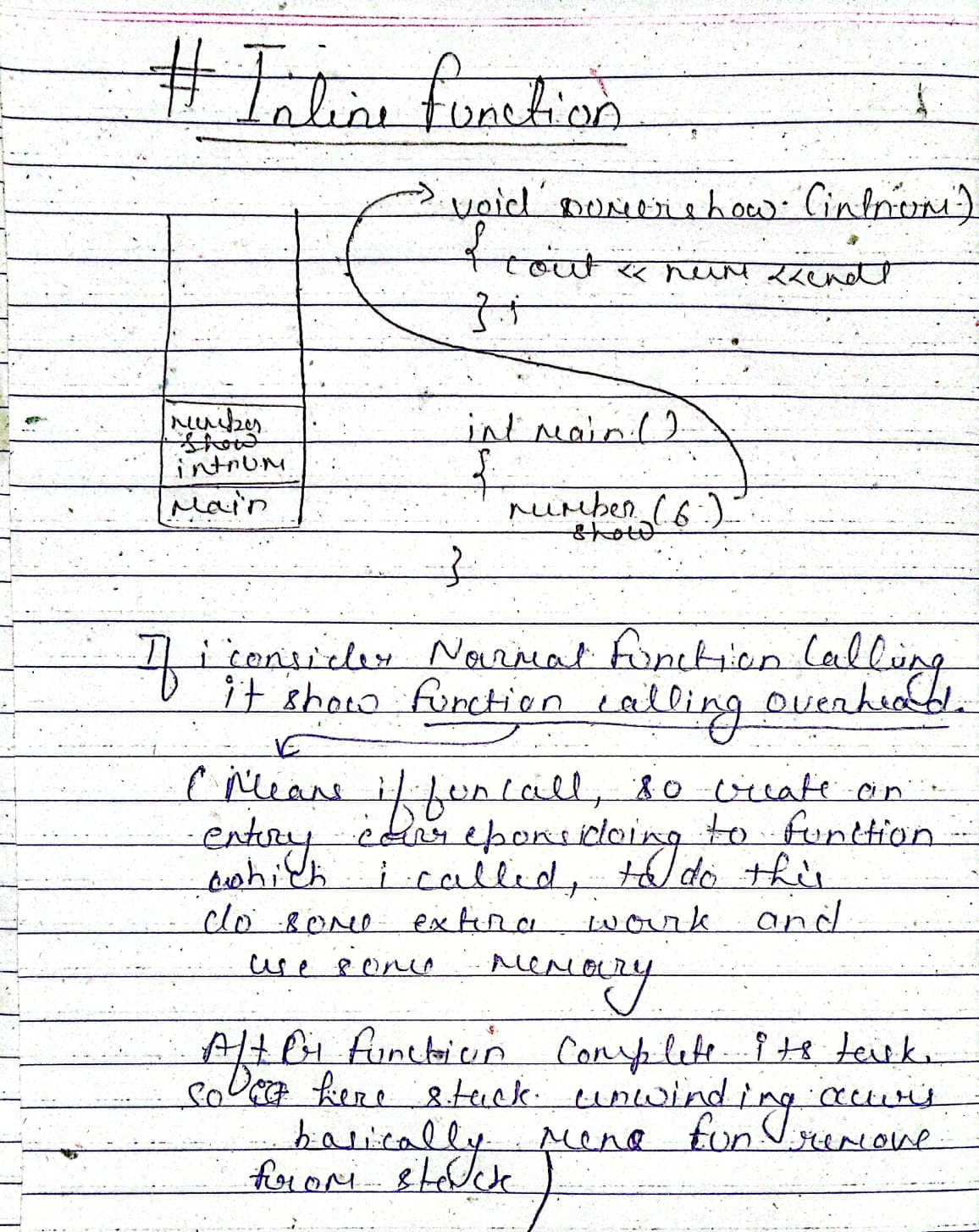
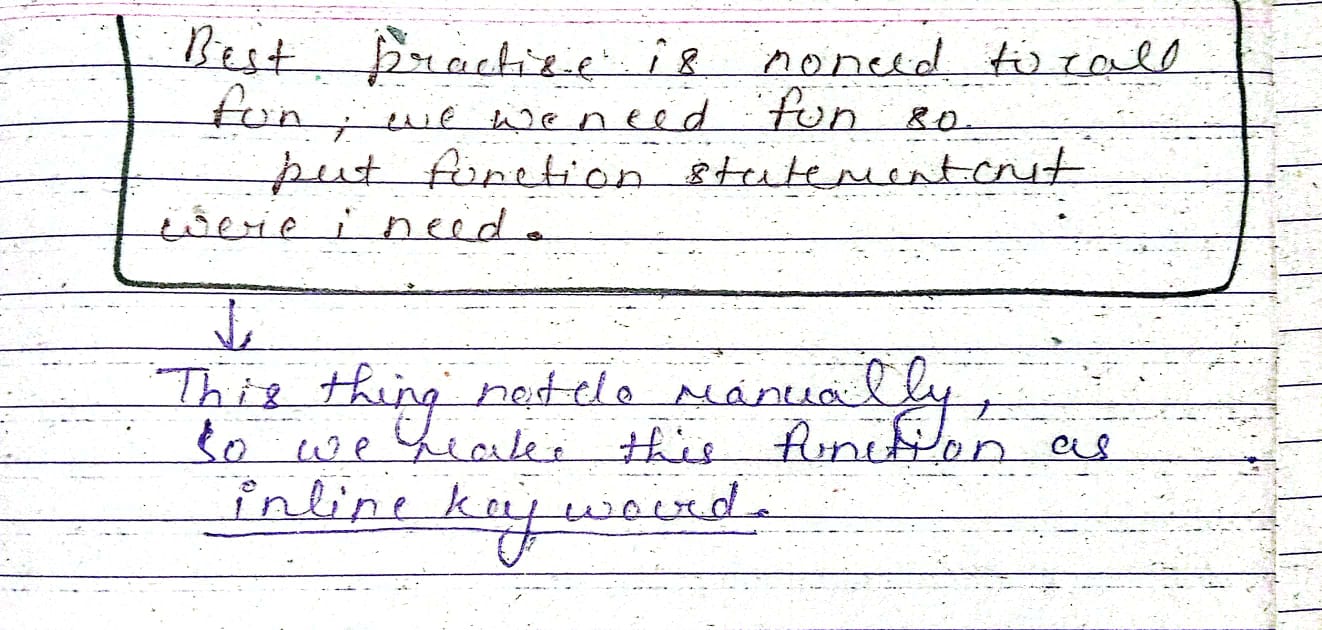
C++ provides inline functions to reduce the function call overhead. An inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of the inline function call. This substitution is performed by the C++ compiler at compile time. An inline function may increase efficiency if it is small.



#include<iostream>

using namespace std;

inline int getMax(int& a, int& b)

{

    return (a>b) ? a : b;   // Ternary Operator

}

int main() {

    int a = 1, b = 2;

    int ans = 0;

    ans = getMax(a,b);

    cout << ans << endl;

    a = a + 3;

    b = b + 1;

    ans = getMax(a,b);

    cout << ans << endl;

    return 0;

}

**The compiler may not perform inlining in such circumstances as:**

1. If a function contains a loop. (*for, while and 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 a function body.
5. If a function contains a switch or goto statement.

**Inline functions Advantages:**

1. Function call overhead doesn’t occur.
2. It also saves the overhead of push/pop variables on the stack when a function is called.
3. It also saves the overhead of a return call from a function.
4. When you inline a function, you may enable the compiler to perform context-specific optimization on the body of the function. Such optimizations are not possible for normal function calls.

**Inline function Disadvantages:**

1. The added variables from the inlined function consume additional registers, After the in-lining function if the variable number which is going to use the register increases then they may create overhead on register variable resource utilization.
2. If you use too many inline functions then the size of the binary executable file will be large, because of the duplication of the same code.
3. Too much inlining can also reduce your instruction cache hit rate, thus reducing the speed of instruction fetch from that of cache memory to that of primary memory.
4. The inline function may increase compile time overhead if someone changes the code inside the inline function then all the calling location has to be recompiled because the compiler would be required to replace all the code once again to reflect the changes, otherwise it will continue with old functionality.