Inline function

There is an important feature in C++, called an inline function, that is commonly used with classes. will make heavy use of it, inline functions are examined here. In C++, you can create short functions that are not actually called; rather, their code is expanded in line at the point of each invocation. This process is similar to using a function-like macro.

To cause a function to be expanded in line rather than called, precede its definition with the inline keyword. For example, in this program, the function max( ) is expanded in line instead of called:

#include <iostream>

using namespace std;

inline int max(int a, int b)

{

return a>b ? a : b;

}

int main()

{

cout << max(10, 20);

cout << " " << max(99, 88);

return 0;

}

The reason that inline functions are an important addition to C++ is that they allow you to create very efficient code. Since classes typically require several frequently executed interface functions (which provide access to private data), the efficiency of these functions is of critical concern. As you probably know, each time a function is called, a significant amount of overhead is generated by the calling and return mechanism. Typically, arguments are pushed onto the stack and various registers are saved when a function is called, and then restored when the function returns. The trouble is that these instructions take time. However, when a function is expanded in line, none of those operations occur. Although expanding function calls in line can produce faster run times, it can also result in larger code size because of duplicated code. For this reason, it is best to inline only very small functions. Further, it is also a good idea to inline only those functions that will have significant impact on the performance of your program.