### Inline Functions

Define functions inline only when they are small, say, 10 lines or less.

**Definition:**

You can declare functions in a way that allows the compiler to expand them inline rather than calling them through the usual function call mechanism.

**Pros:**

Inlining a function can generate more efficient object code, as long as the inlined function is small. Feel free to inline accessors and mutators, and other short, performance-critical functions.

**Cons:**

Overuse of inlining can actually make programs slower. Depending on a function's size, inlining it can cause the code size to increase or decrease. Inlining a very small accessor function will usually decrease code size while inlining a very large function can dramatically increase code size. On modern processors smaller code usually runs faster due to better use of the instruction cache.

**Decision:**

A decent rule of thumb is to not inline a function if it is more than 10 lines long. Beware of destructors, which are often longer than they appear because of implicit member- and base-destructor calls!

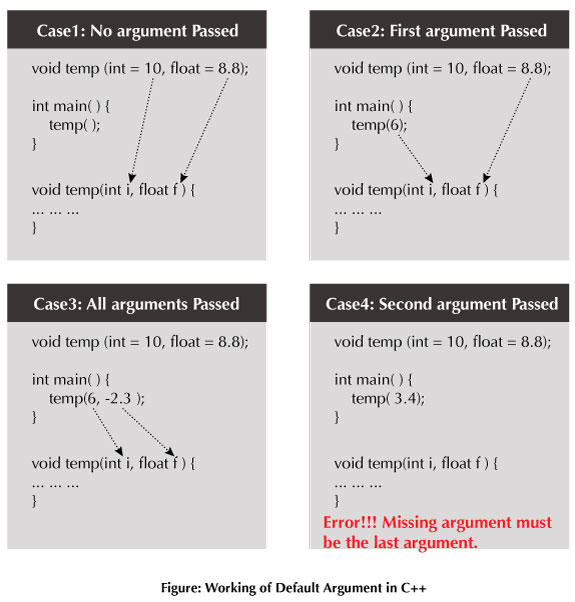
Another useful rule of thumb: it's typically not cost effective to inline functions with loops or switch statements (unless, in the common case, the loop or switch statement is never executed).

It is important to know that functions are not always inlined even if they are declared as such; for example, virtual and recursive functions are not normally inlined. Usually recursive functions should not be inline. The main reason for making a virtual function inline is to place its definition in the class, either for convenience or to document its behavior, e.g., for accessors and mutators.

**Reference:** https://google-styleguide.googlecode.com/svn/trunk/cppguide.html#Inline\_Functions

**Functions Overloading in C++: Working with Default Arguments**

In C++ programming, you can provide default values for function parameters. The idea behind default argument is very simple. If a function is called by passing argument/s, those arguments are used by the function. But if all argument/s are not passed while invoking a function then, the default value passed to arguments are used. Default value/s is passed to argument/s in function prototype. Working of default argument is demonstrated in the figure below:



But pass to the first argument is 3

**Example: Default Argument**

|  |  |
| --- | --- |
| //Second argument passed but passed to the first argument is 3  #include<iostream>  using namespace std;  void add(int i = 3, float j = 2.5);  int main()  {  cout<<"\n";  add(3.4);  }  void add(int i, float j)  {  cout<<(i+j);  } | **5.5** |

/\* C++ Program to demonstrate working of default argument \*/

|  |
| --- |
| #include <iostream>  using namespace std;  void display(char = '\*', int = 1);  int main() {  cout<<"No argument passed:\n";  display();  cout<<"\n\nFirst argument passed:\n";  display('#');  cout<<"\n\nSecond argument passed:\n";// but here pass first argument again. See note 1.  display (65);// There is no ERROR. Because the int is automatically converted to char type.  cout<<"\n\nBoth argument passed:\n";  display('$', 5);  return 0;  }  void display(char c, int n){  for(int i = 1; i <=n; ++i) {  cout<<c;  }  cout<<endl;  } |

**Output:**

|  |
| --- |
| No argument passed:  \*  First argument passed:  #  Second argument passed:  A  Both argument passed:  $$$$$ |

In the above program, you can see default value passed to arguments (in function prototype). At first, display () function is called without passing any arguments. In this case, default () function used both default arguments. Then, the function is called using only first argument. In this case, function does not use first default value passed. Function uses the actual parameter passed as first argument and takes default value (second value in function prototype) as its second argument. When display () is invoked passing both arguments, default arguments are not used.

**Note 1: The missing argument must be the last argument of the list, that is, if you are passing only one argument in the above function, it should be the first argument.**

**Reference:**

1. http://www.programiz.com/cpp-programming/default-argument