

# Inline Functions, Default Arguments & Constant Arguments in C++ | C++ Tutorials for Beginners #17

In this tutorial, we will discuss inline functions, default arguments, and constant arguments in C++

## Inline Functions in C++

Inline functions are used to reduce the function call. When one function is being called multiply times in the program it increases the execution time, so inline function is used to reduce time and increase program efficiency. If the inline function is being used when the function is called, the inline function expands the whole function code at the point of a function call, instead of running the function. Inline functions are considered to be used when the function is small otherwise it will not perform well. Inline is not recommended when static variables are being used in the function. An example of an inline function is shown in Code Snippet 1.

```
inline int product(int a, int b){
    return a*b;
}
```

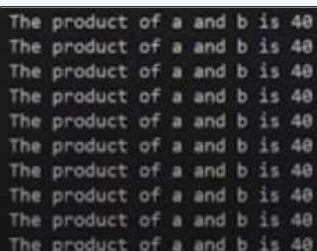
**Code Snippet 1: Inline function**

As shown in Code Snippet 1, 1<sup>st</sup> inline keyword is used to make the function inline. 2<sup>nd</sup> a product function is created which has two arguments and returns the product of them. Now we will call the product function multiple times in our main program which is shown in Code Snippet 2.

```
int main(){
    int a, b;
    cout<<"Enter the value of a and b"<<endl;
    cin>>a>>b;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    cout<<"The product of a and b is "<<product(a,b)<<endl;
    return 0;
}
```

**Code Snippet 2: Calling Inline Product Function**

As shown in Code Snippet 2, we called the product function multiple times. The main thing to note here is that the function will not run instead of it the function code will be copied at the place where the function is being called. This will increase the execution time of the program because the compiler doesn't have to copy the values and get the return value again and again from the compiler. The output of the following program is shown in figure 1.



```
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
The product of a and b is 40
```

**Figure 1: Inline Function Output**

## Default Arguments in C++

Default arguments are those values which are used by the function if we don't input our value. It is recommended to write default arguments after the other arguments. An example program for default arguments is shown in Code Snippet 3.

```
float moneyReceived(int currentMoney, float factor=1.04){
    return currentMoney * factor;
}

int main(){
    int money = 100000;
    cout<<"If you have "<<money<<" Rs in your bank account, you will receive "<<moneyReceived(money)<<" Rs after 1 year<<endl;
    cout<<"For VIP: If you have "<<money<<" Rs in your bank account, you will receive "<<moneyReceived(money, 1.1)<<" Rs after 1 year<<endl;
    return 0;
}
```

#### Code Snippet 3: Default Argument Example Program

As shown in Code Snippet 3, we created a “moneyReceived” function which has two arguments “int currentMoney” and “float factor=1.04”. This function returns the product of “currentMoney” and “factor”. In our main function, we called “moneyReceived” function and passed one argument “money”. Again we called “moneyReceived” function and passed two arguments “money” and “1.1”. The main thing to note here is that when we passed only one argument “money” to the function at that time the default value of the argument “factor” will be used. But when we passed both arguments then the default value will not be used. The output for the following program is shown in figure 2.

```
PS D:\Business\code playground\C++ course> cd "D:\Business\code playground\C++ course\" ; if ($?) { g++ tut17.cpp -o tut17 } ; if ($?) { .\tut17 }
If you have 100000 Rs in your bank account, you will receive 104000Rs after 1 yearFor VIP: If you have 100000 Rs in your bank account, you will receive 110000Rs af
ter 1 year
```

#### Figure 2: Default Argument Example Program Output

## Constant Arguments in C++

Constant arguments are used when you don't want your values to be changed or modified by the function. An example of constant arguments is shown in Code Snippet 4.

```
int strlen(const char *p){
    // ...
}
```

#### Code Snippet 4: Constant Arguments Example

As shown in Code Snippet 4, we created a “strlen” function which takes a constant argument “p”. As the argument is constant so its value won't be modified.

## Code as described/written in the video

```
#include<iostream>
using namespace std;

inline int product(int a, int b){
    // Not recommended to use below lines with inline functions
    // static int c=0; // This executes only once
    // c = c + 1; // Next time this function is run, the value of c will be retained
    return a*b;
}

float moneyReceived(int currentMoney, float factor=1.04){
    return currentMoney * factor;
}

// int strlen(const char *p){

// }
int main(){
    int a, b;
    // cout<<"Enter the value of a and b"<<endl;
    // cin>>a>>b;
    // cout<<"The product of a and b is "<<product(a,b)<<endl;
    int money = 100000;
    cout<<"If you have "<<money<<" Rs in your bank account, you will receive "<<moneyReceived(money)<<"Rs aft
```

[← Previous](#)[Next →](#)