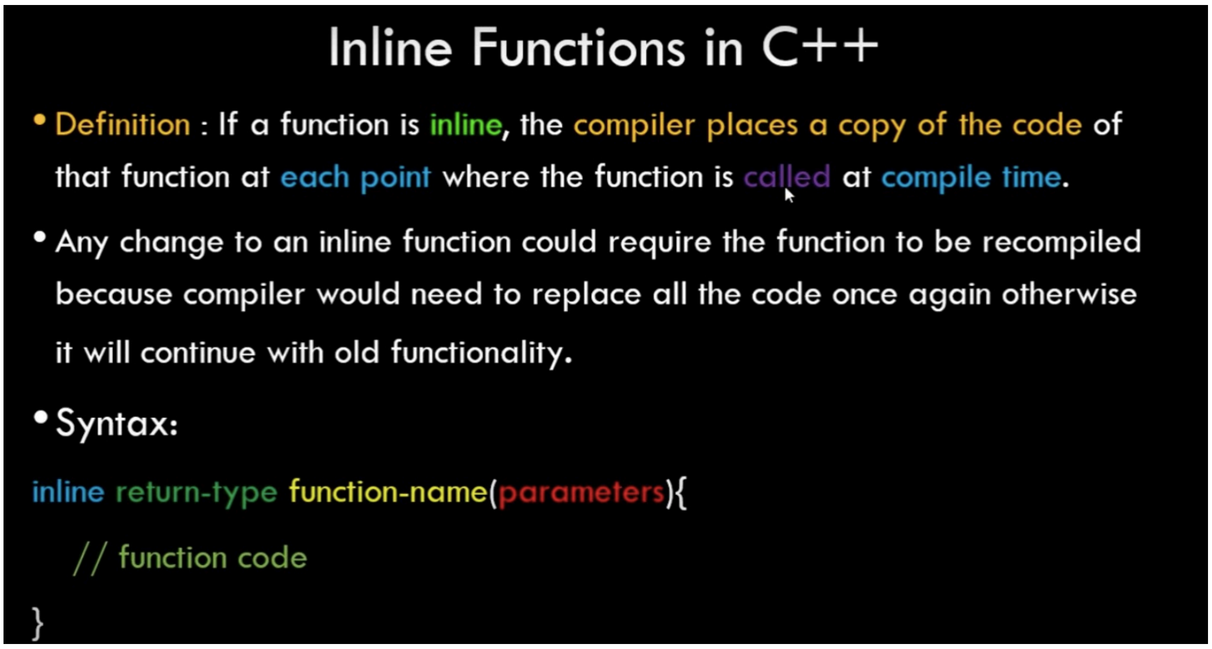
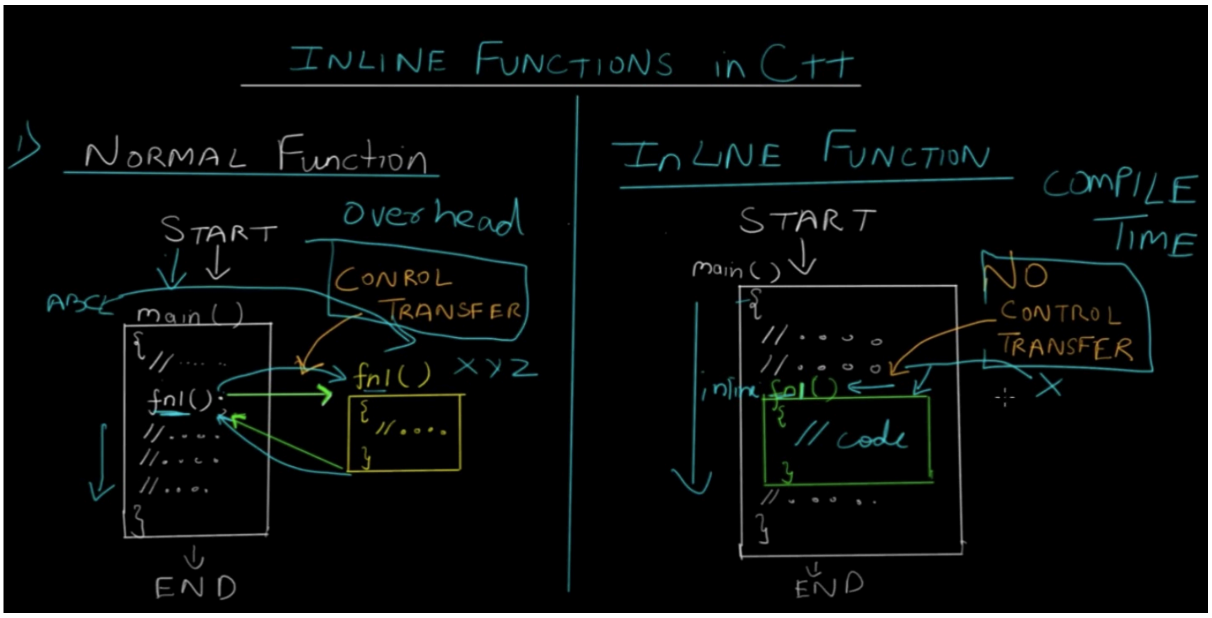
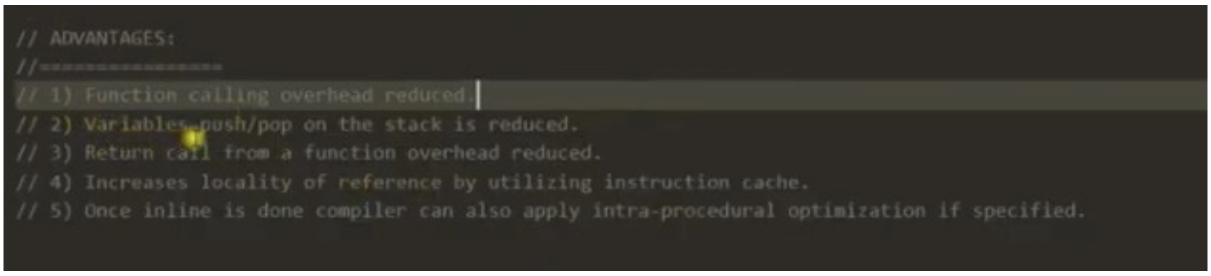


Inline Functions in C++

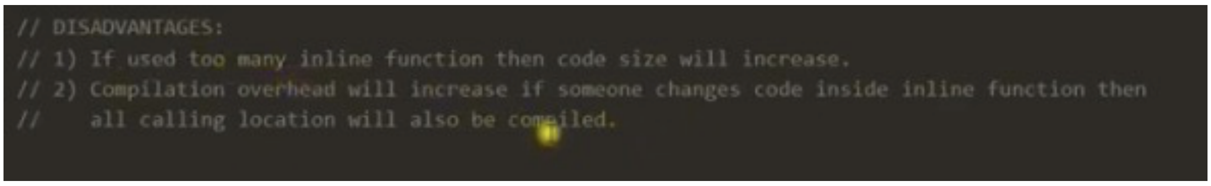
- Definition : If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.
- Any change to an inline function could require the function to be recompiled because compiler would need to replace all the code once again otherwise it will continue with old functionality.
- Syntax:

```
inline return-type function-name(parameters){
    // function code
}
```

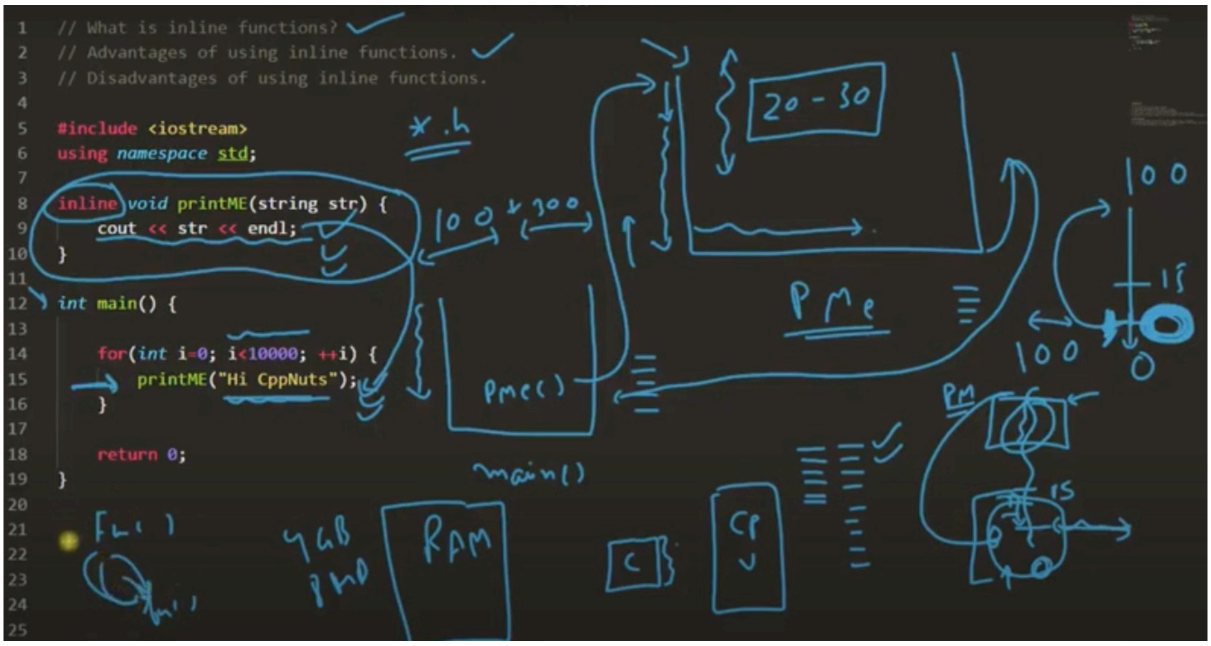




```
// ADVANTAGES:
//================
// 1) Function calling overhead reduced.
// 2) Variables push/pop on the stack is reduced.
// 3) Return call from a function overhead reduced.
// 4) Increases locality of reference by utilizing instruction cache.
// 5) Once inline is done compiler can also apply intra-procedural optimization if specified.
```



```
// DISADVANTAGES:
// 1) If used too many inline function then code size will increase.
// 2) Compilation overhead will increase if someone changes code inside inline function then
//    all calling location will also be compiled.
```



```
// What is inline functions?
// Advantages of using inline functions.
// Disadvantages of using inline functions.

#include <iostream>
using namespace std;

inline void printME(string str) {
    cout << str << endl;
}

int main() {

    for(int i=0; i<10000; ++i) {
        printME("Hi CppNuts");
    }

    return 0;
}
```

Code: 1

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

inline int add(int a, int b)
{
        return (a+b);
}


int main()
{

         for(int i=0;i<100;i++)
        {
  cout<<"Additon of 4 & 5is:"<<add(4,5)<<endl;
        }
return 0;
}
```

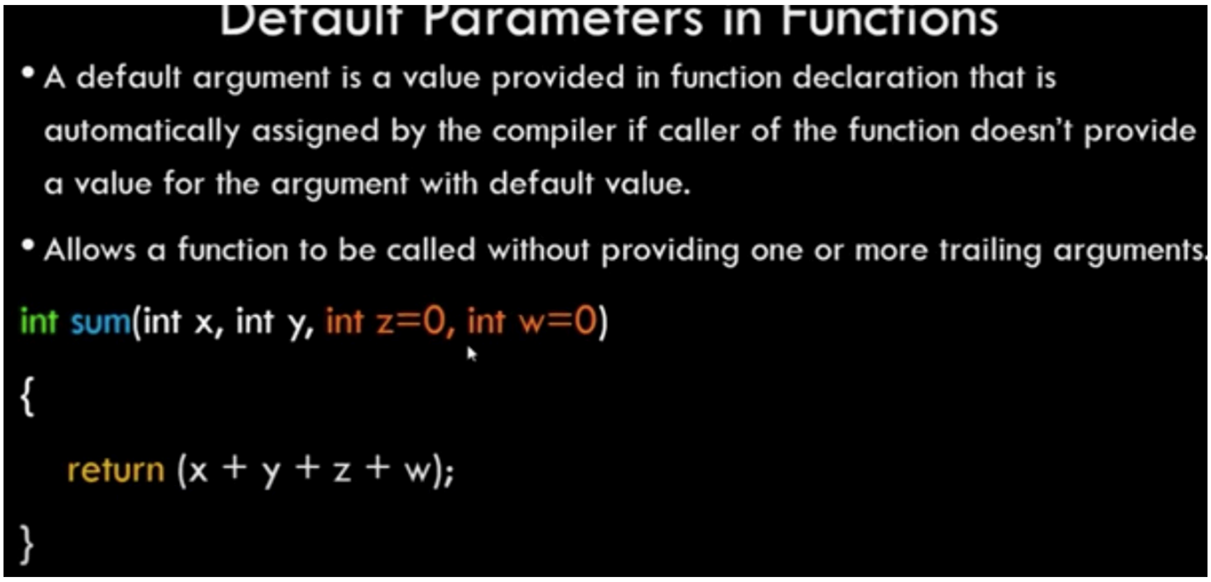
Code 2:

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

inline void display(string str)
{
        cout<< str << endl;
}

int main()
{
        for(int i=0;i<100;i++)
        {
 display("Hello 100 times"); // cout<< str << endl;
        }
return 0;
}
```

Default Parameters in C++



Default Parameters in Functions

- A default argument is a value provided in function declaration that is automatically assigned by the compiler if caller of the function doesn't provide a value for the argument with default value.
- Allows a function to be called without providing one or more trailing arguments.

```
int sum(int x, int y, int z=0, int w=0)
{
    return (x + y + z + w);
}
```

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

int sum(int a, int b, int c=0, int d=5)
{
    return(a+b+c+d);
}

int main()
{
    cout<<"Sum of 1,2,3,4 is:"<<sum(1,2,3,4);
    cout<<"Sum of 1,2 is:"<<sum(1,2);
    return 0;
}
```