unacademy

Inline Function

Comprehensive Course On C++

Pankaj Sharma • Lesson 3 • May 14, 2024

CS & IT Engineering

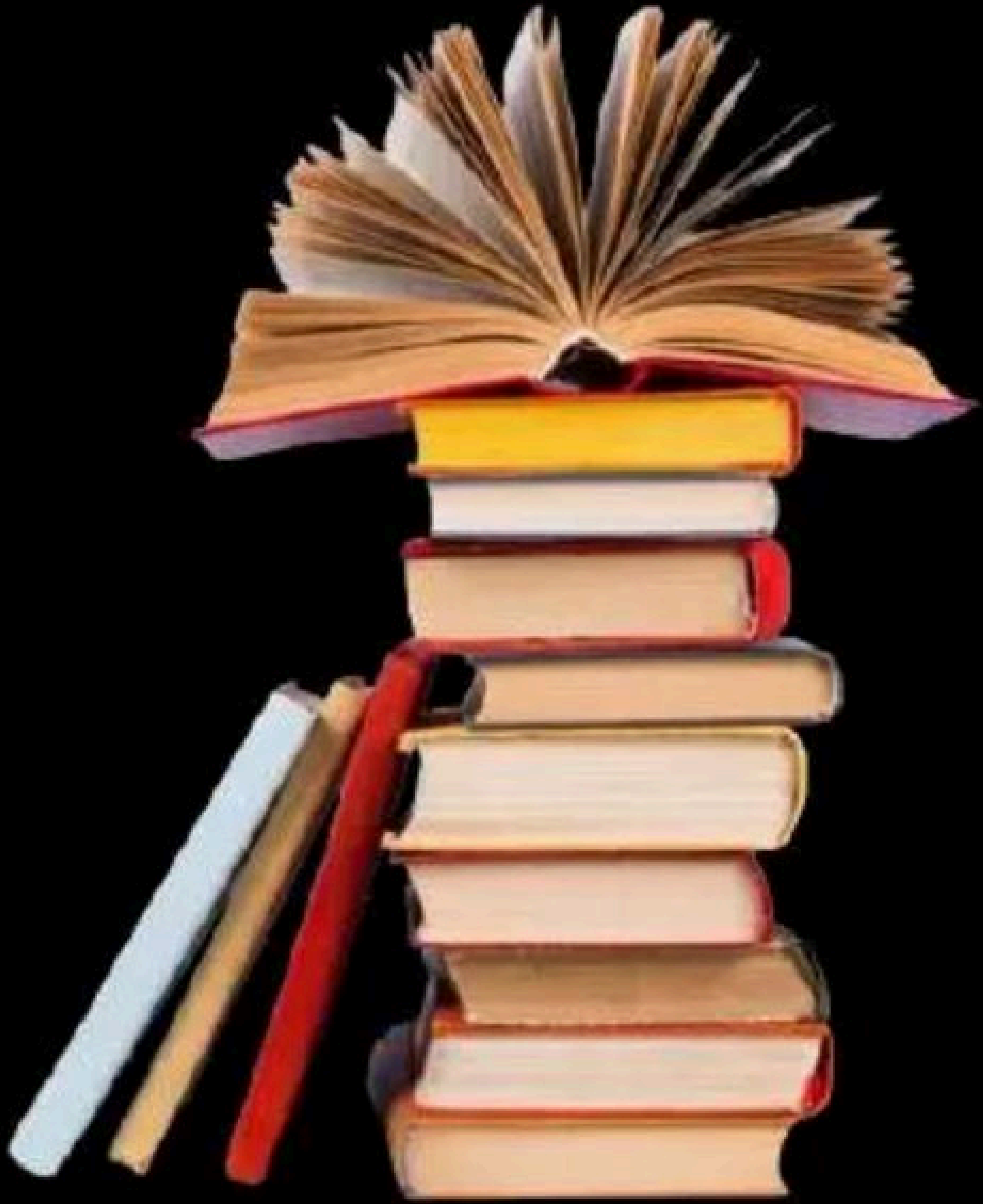
C++

Inline Functions

Lecture - 03

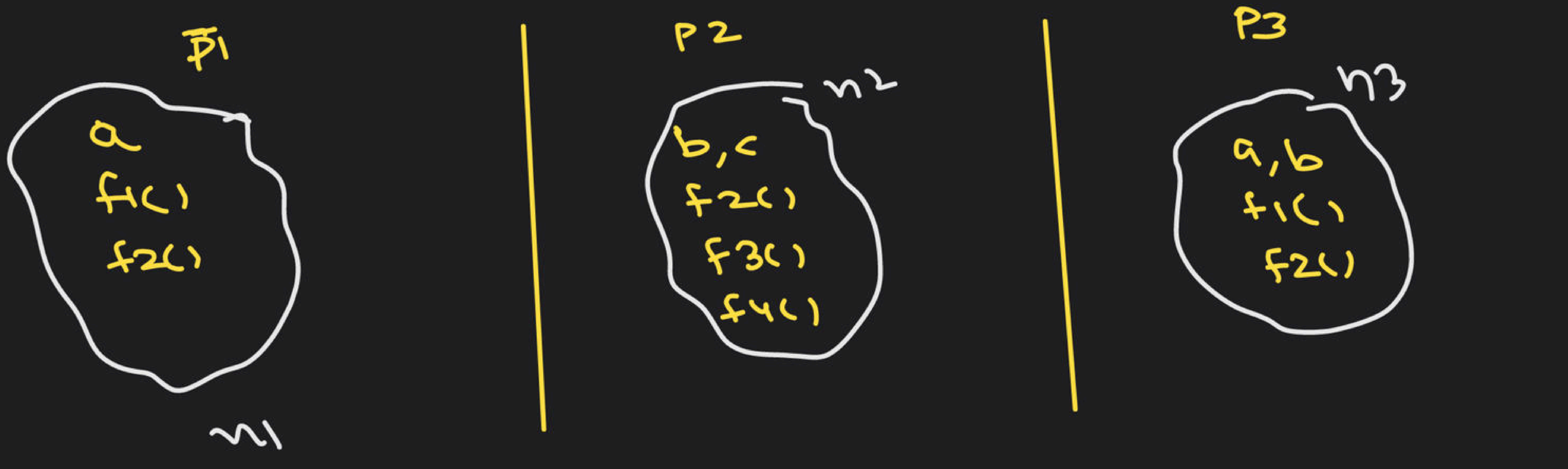
By- Pankaj Sir

Topics *to be covered*



1. Inline functions

namespace



n1::f1() ✓   n2::b ✓

functions

Adv.

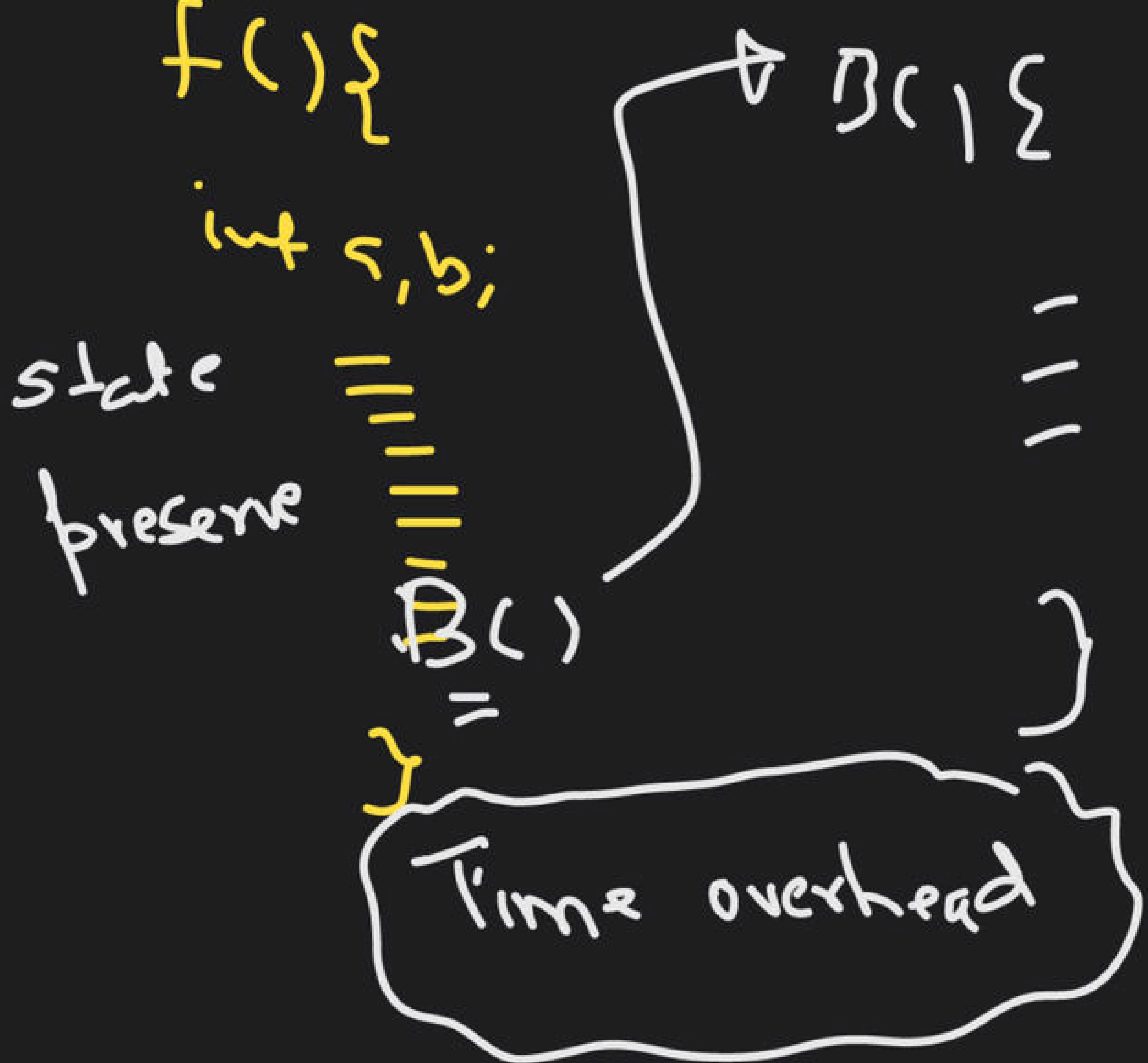
① Reusability of code

② Easy to read, modify.

③ Memory:

Saves memory

dis.



```
f(){
    int a,b;
    =
    =
    =
    =
    =
    =
    B()
    =
}
```

```
B(){
    =
    =
    =
}
```

Memory save but exe time inc. } Sol ×

① $\boxed{\text{Statement}}$ } less amount of memory ✓

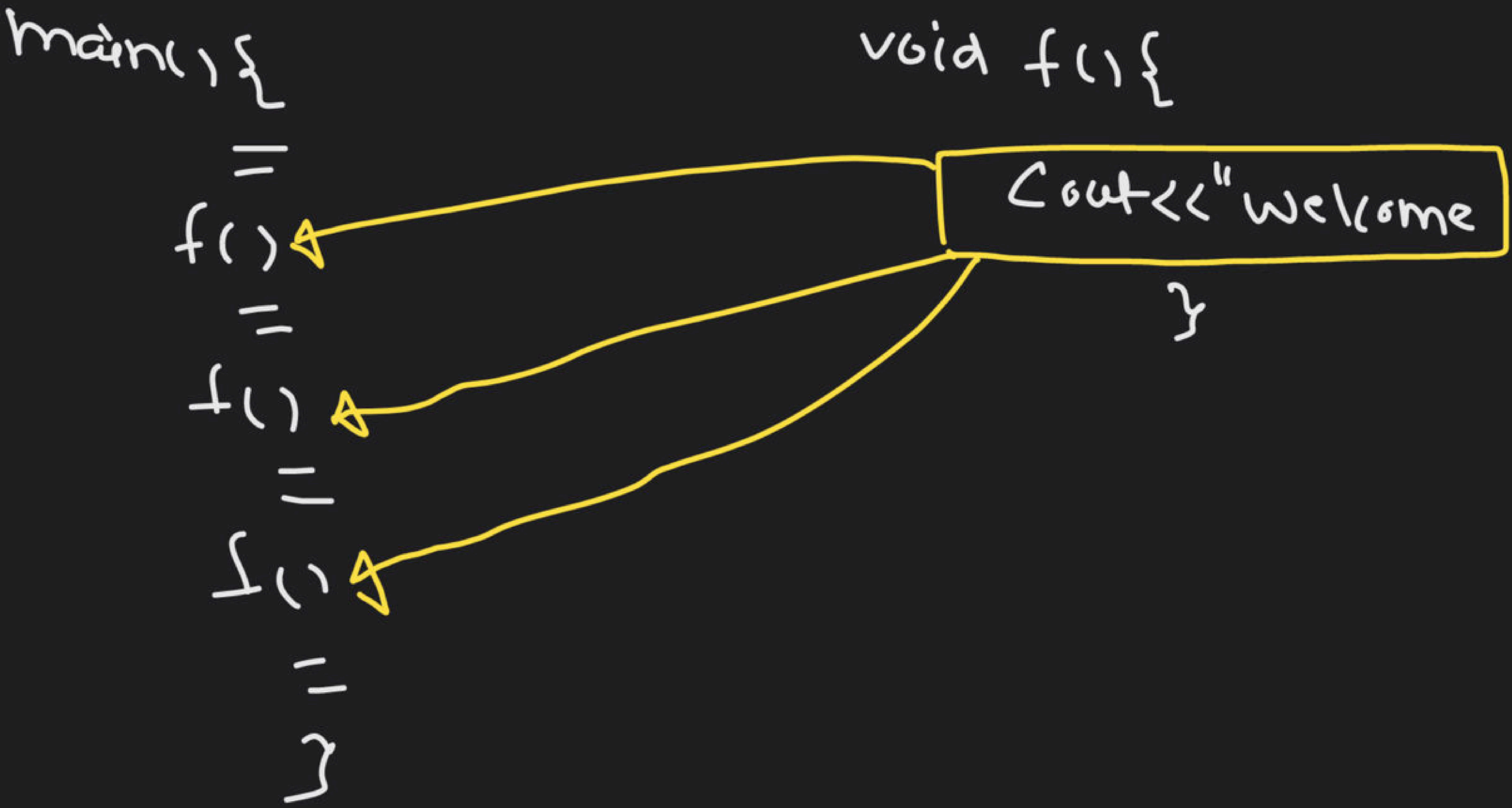
→ f( ) {
statement
}

② $\boxed{\begin{array}{l}\text{statement 1} \\ \text{statement 2} \\ \vdots \\ \text{statement 100}\end{array}}$ → g( ) {
Statem1
⋮
Statement100
}

$$\begin{tikzcd}
\text{Code} \arrow[r] \arrow[d, bend right] & \text{less amount of memory} \\
\text{func} \arrow[r] & \text{adv} \times
\end{tikzcd}$$

Inline function

'inline'

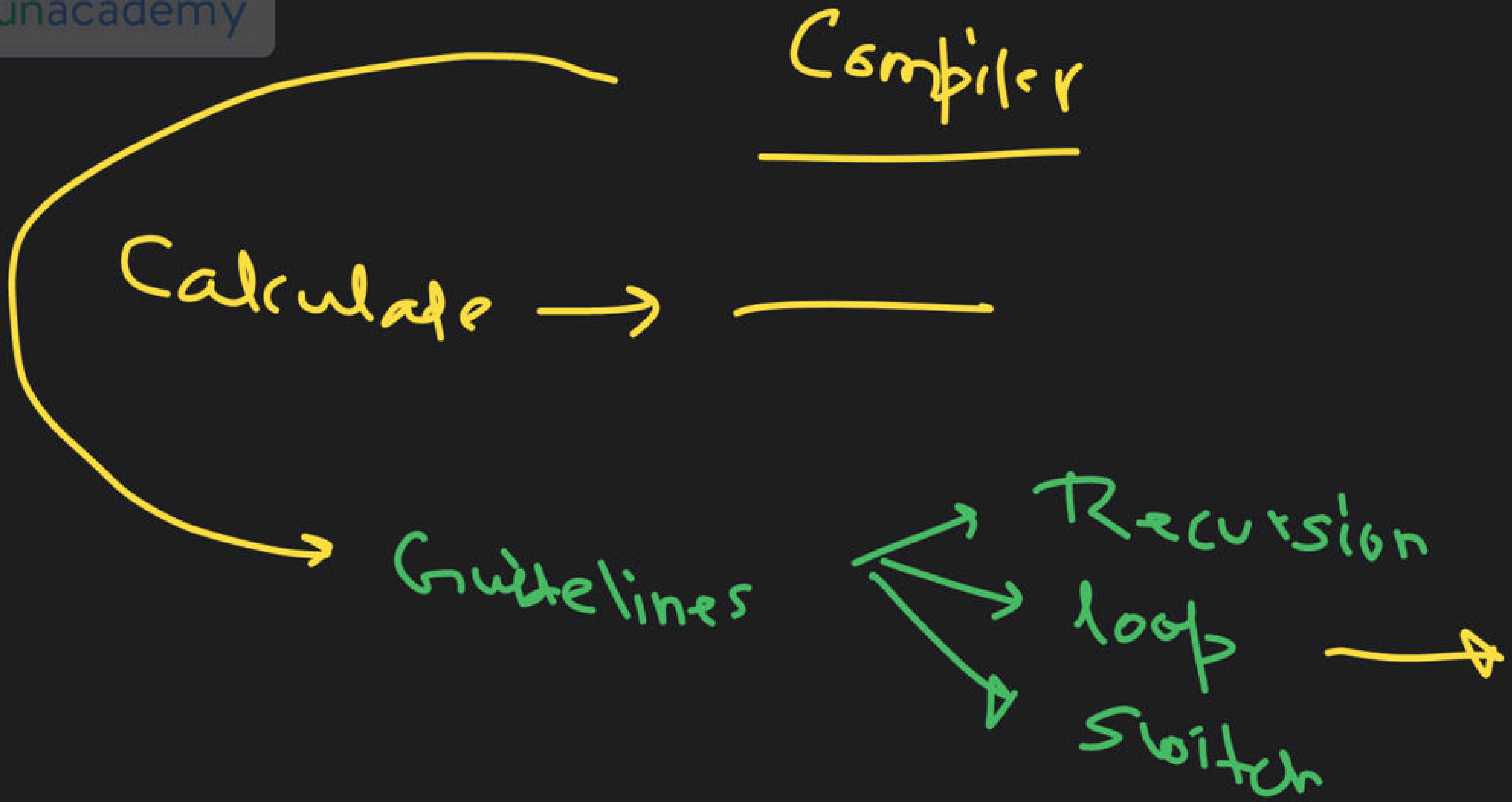


```
main(){
  =
  f()
  =
  f()
  =
  f()
  =
}
```

```
void f(){
  cout<<"welcome
}
```

```
#include<iostream>
using namespace std;
int square(int);
void main(){
        int n, ans;
        Cout<<"Enter a no";
        cin>>n;
        ans= square(n);
        Cout<<"The square is"<<ans;
    }
int squar(int a) {
        return a*a;
```

$$\begin{tikzcd} \boxed{\text{inline}} \arrow[d] \\ \text{Compiler} \end{tikzcd}$$


Compiler
Calculate
Guidelines
Recursion
loop
Switch

inline

Programmer

Compiler

Advantage ?

$$\text{Programmer} \longrightarrow \text{source code}$$

func adv

$$\text{Compiler} \longrightarrow \text{obj}$$

func. adv. ✓

dis ✗

1.) inline is a keyword used for optimization purpose.

2.) inline is a request to compiler.

Default Arguments

```
#include<iostream>
int Add(int,int);
using namespace std;
void main(){

    int x=10, y=20;

    cout<< Add(x,y);
    cout<< Add(10,20,30);
}
```

recieve exactly 2 arg

Error

```
int Add(int a, int b)
{
    return a+b;
}
```

C++ $\longrightarrow$ $\boxed{\text{Default arguments}}$

function $\longrightarrow$ 3 arg. dec.

Add(10, 20) $\longrightarrow$ $\underline{\underline{\text{Error}}}$

2 arg. must be there

$\overline{\text{int add(int, int,}}$ $\boxed{\text{int = 0}}$) ;

→ optional

if we don't provide

⇒ default value is taken

✓ add(10, 20); ⇒ 3rd arg → 0

✓ add(10, 20, 30);

add(10); ⟶ Error

```
int add( int, int = 0, int = 0);
```

atleast 1 arg.

default arg.

```
add(10);        ✓
add(10,20);     ✓
add(10,20,30);  ✓
```

```
int add(int =20, int =0, int = 10);

int main(){

    add();          ✓
    add(10);        ✓
    add(10,20);     ✓
    add(10,20,30);  ✓
}
```

** Default arguments ⇒ right aligned

```
int add(int=0, int, int);    ✗
int add(int, int=0, int);    ✗
int add(int, int, int=0);    ✓
int add(int, int=0, int=0);  ✓
```

```
add(10,20)
```

* Default arguments can be provided either in dec. or in def.

but not in both.

```
int add(int, int, int=0);
```

1.) object, class

2.) cout, cin, std

3.) inline functions

4.) Default arguments

$\Rightarrow$ refrence variable

```
int add(int x, int y = 10, int z = 30)
{

}
```

unacademy

THANK YOU!

Here's to a cracking journey ahead!