

CS & IT ENGINEERING

Programming in C

Functions and Storage Classes
Lec- 02



By- Pankaj Sharma sir

A stylized graphic of a laptop screen with a blue border and an orange base. The screen displays the text 'TOPICS TO BE COVERED' in a dark teal, sans-serif font.

TOPICS TO BE COVERED

A dashed orange line with arrowheads at both ends, connecting the laptop screen to the 'Storage Classes' box.

Storage Classes

Storage class

- ① Scope : part of code in which a variable is visible.
- ② Lifetime: Duration (Active/Alive)
- ③ default value : If we don't initialize a variable then what is its default value.
- ④ Storage Area : Where a variable is stored.

```
#include <stdio.h>
void main() {
```

```
    int i;
```

```
    printf("%d", i);
}
```

Garbage value

① auto

```
#include <stdio.h>
int Add(int, int);
void main() {
```

```
    int a=10, b=20, sum;
```

```
    sum = Add(a, b);
```

```
    printf("%d", sum);
```

```
}
```

formal arg.

```
int Add(int x, int y)
```

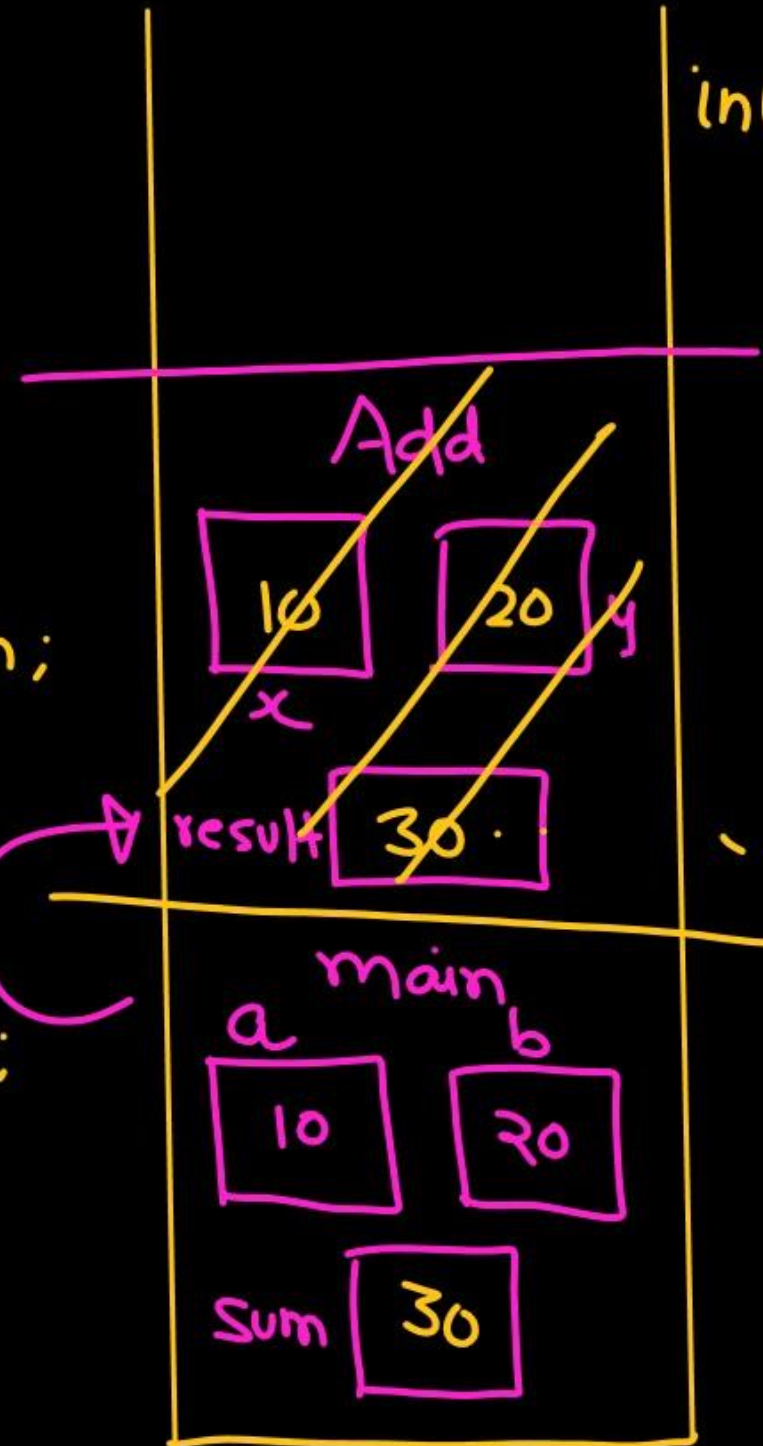
```
{
```

```
    int result;
```

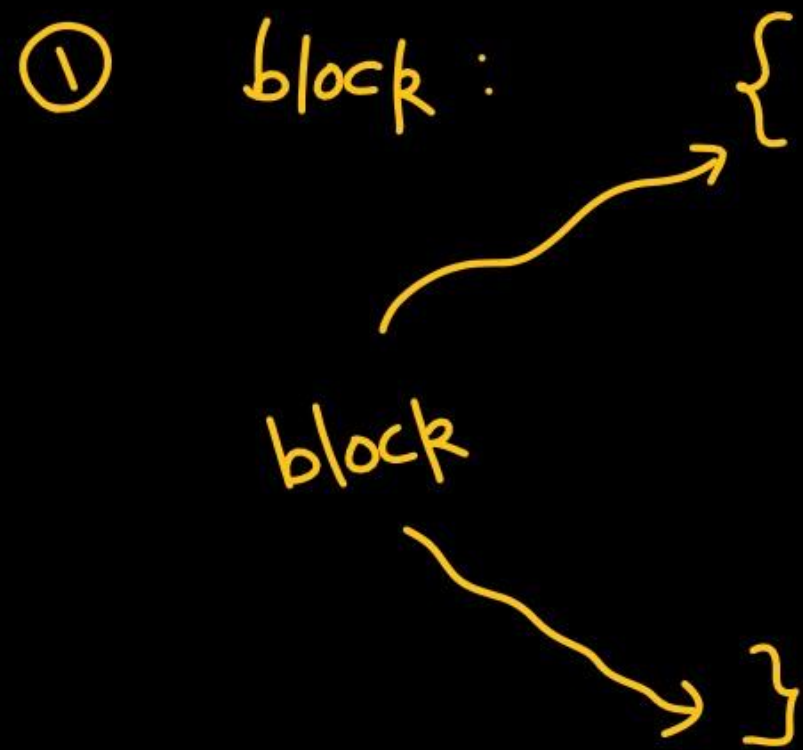
```
    result = x + y;
```

```
    return result;
```

```
}
```



STACK



② File

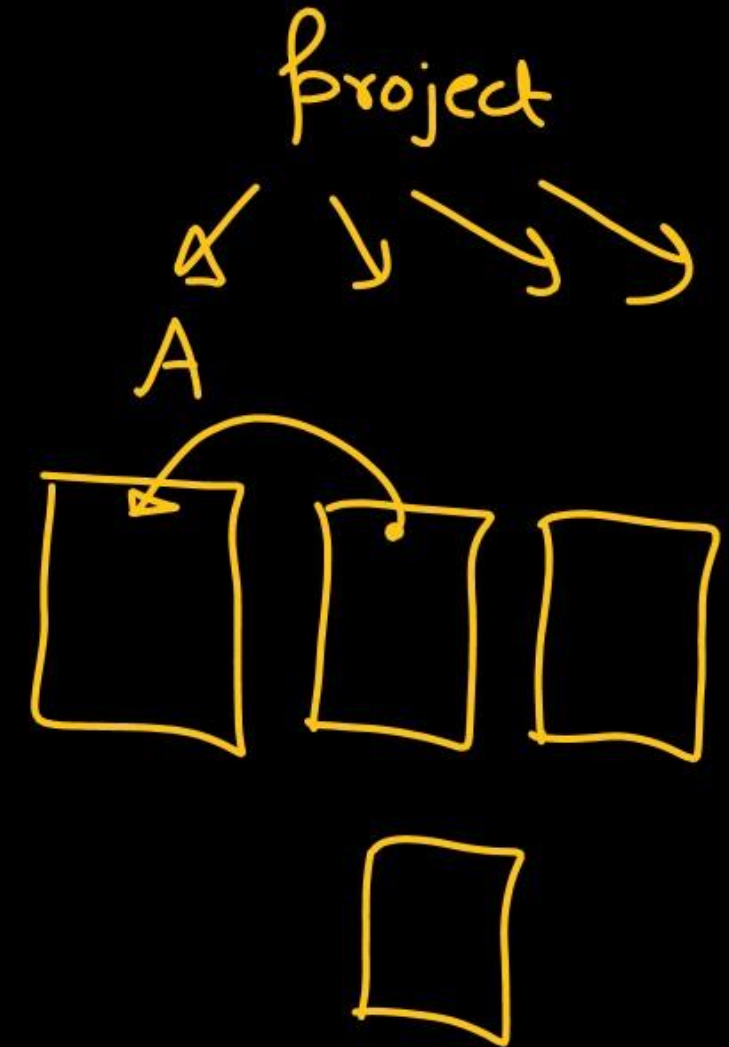
```
void f1(){
    == int z;
}

void f2(){
    = int y;
}

void main(){
    int x;
}
```

Pankaj.c

③ Multiple files




```
void fun1(){
```

```
    int x=10,y=20;
```

```
    printf("%.d",x+y);
```

```
}
```

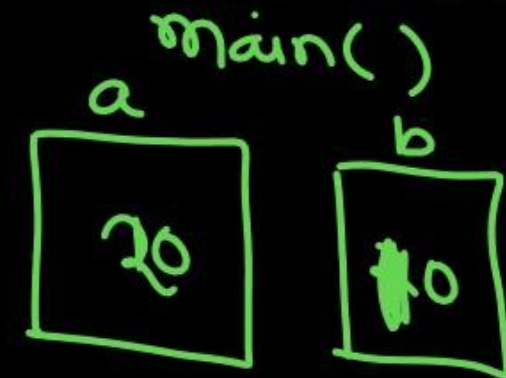
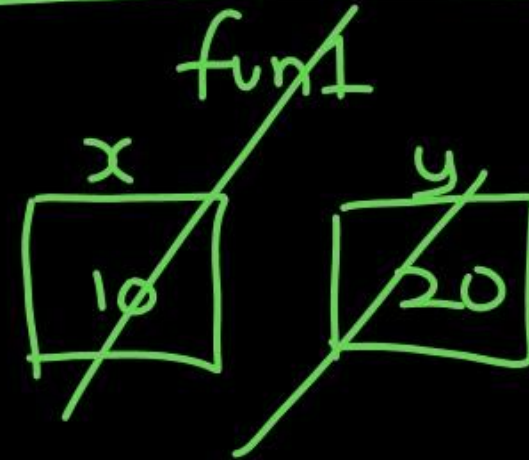
```
void main(){
```

```
    int a=20,b=10;
```

```
    printf("%.d",a+b);
```

```
    fun1();
```

```
}
```



```
void fun1(){
```

```
    int a = 10, b = 20;
```

```
    printf("/d", a+b);
```

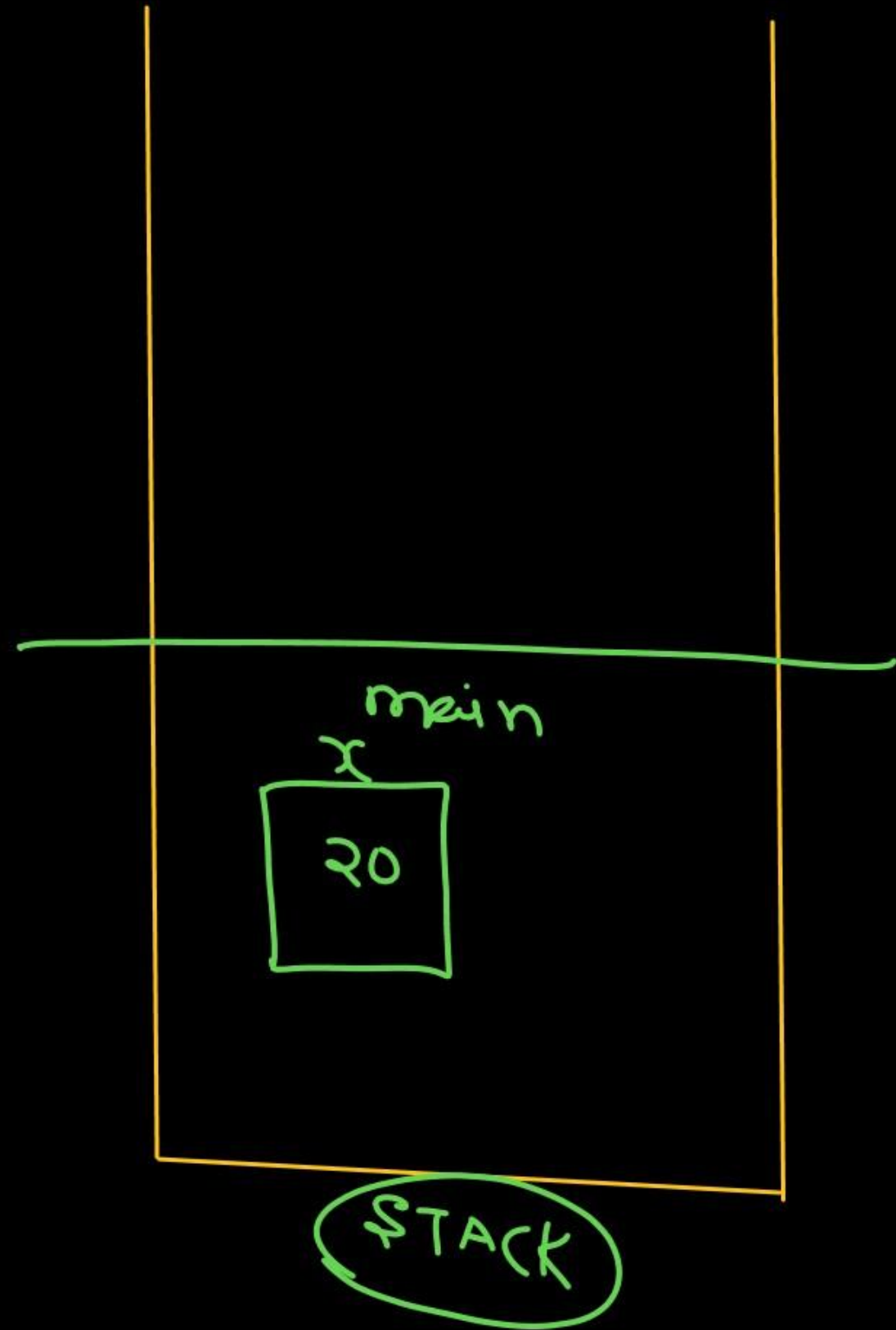
```
}
```

```
void main(){
```

```
    int x = 20;
```

```
    printf("/d", a+x);
```

```
}
```



auto

by default variable declared inside a function are auto.

```
void main(){
```

```
int a=10, b=20;
```

```
}  
=  
}
```

```
void main(){
```

```
auto int a=10, b=20;
```

Same

Optional

scope : block in which they are declared.

Lifetime : block in which they are declared.

default value : Garbage

store : stack

```
void main(){
```

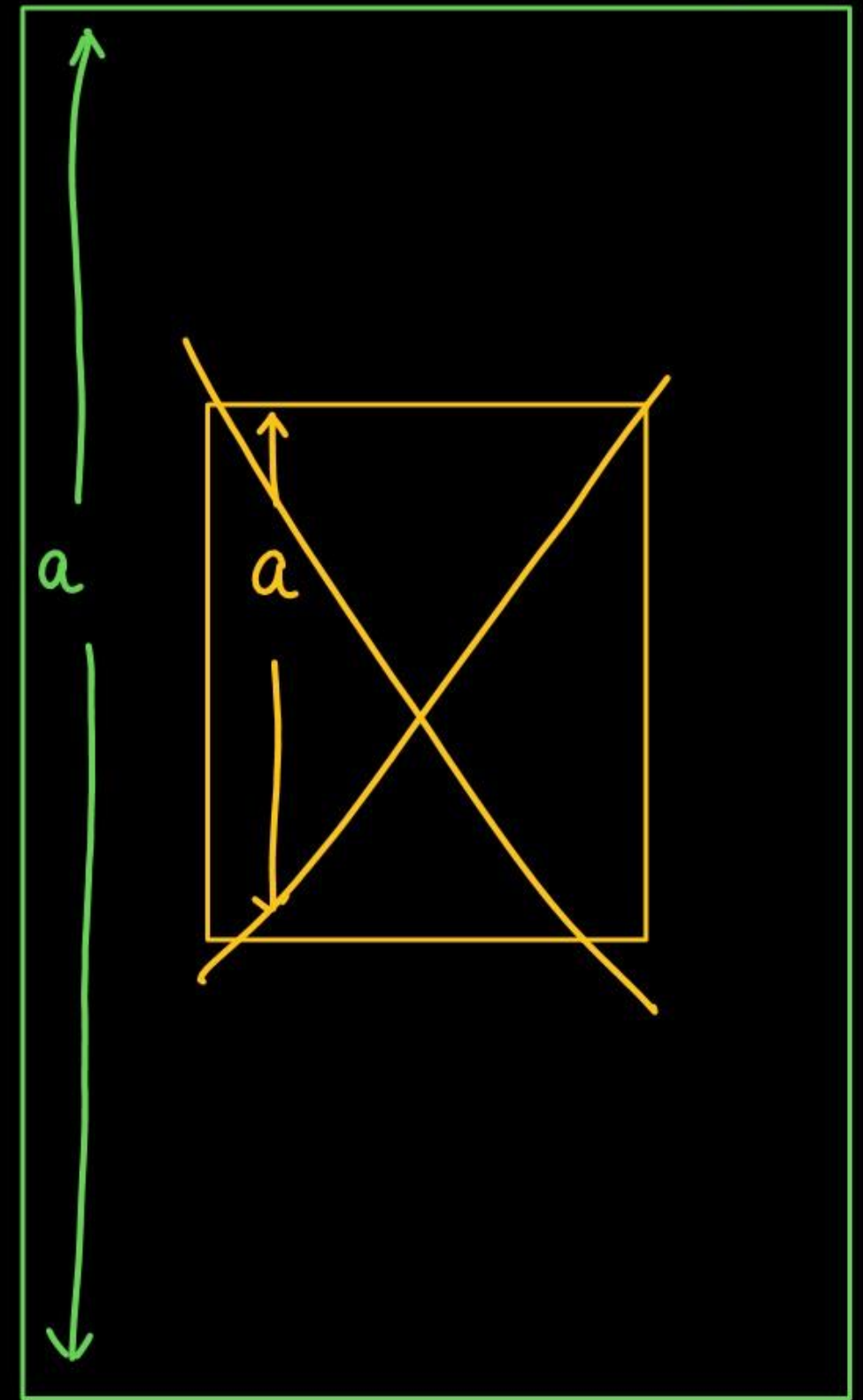
```
    int a ;
```

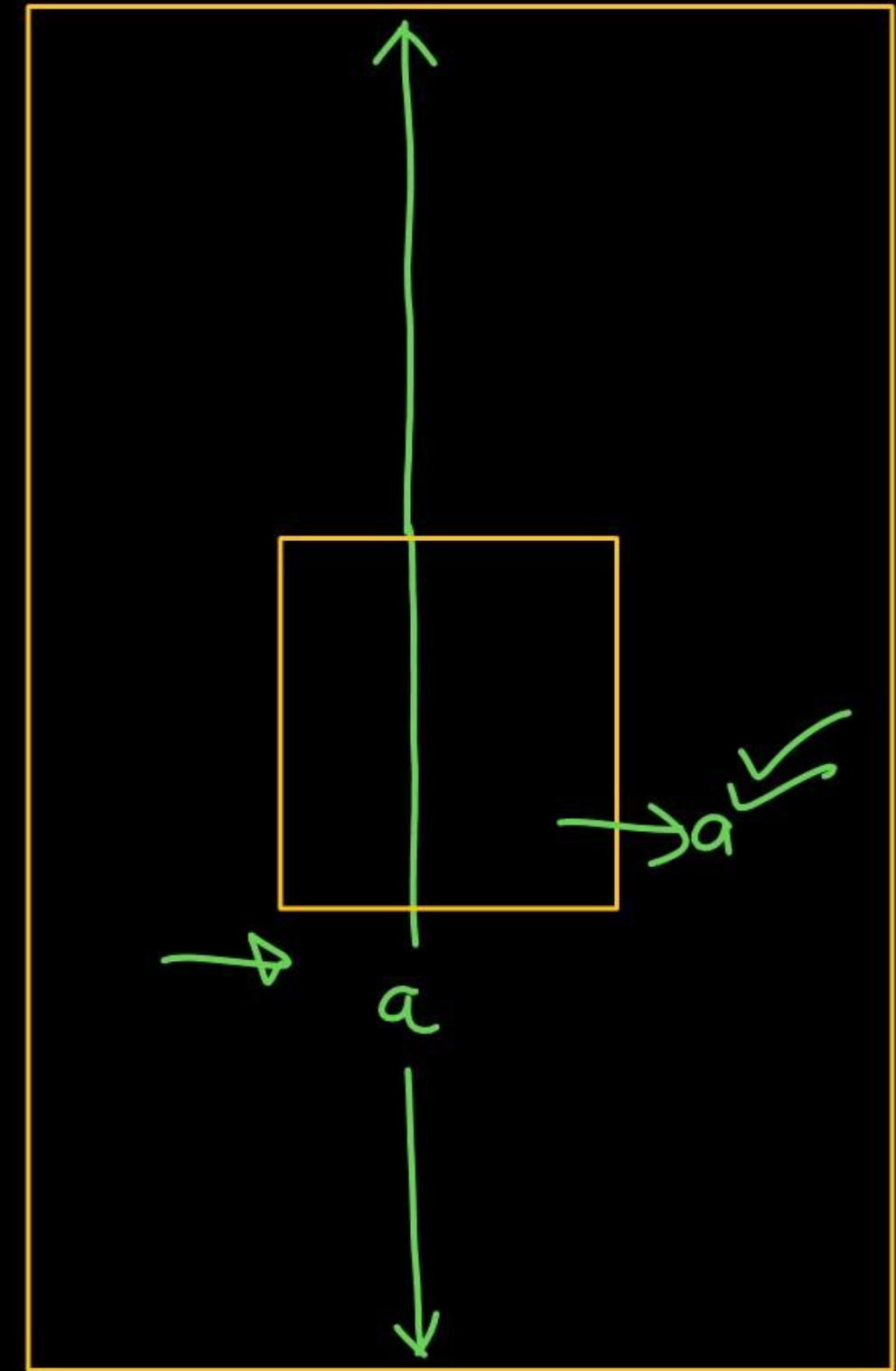
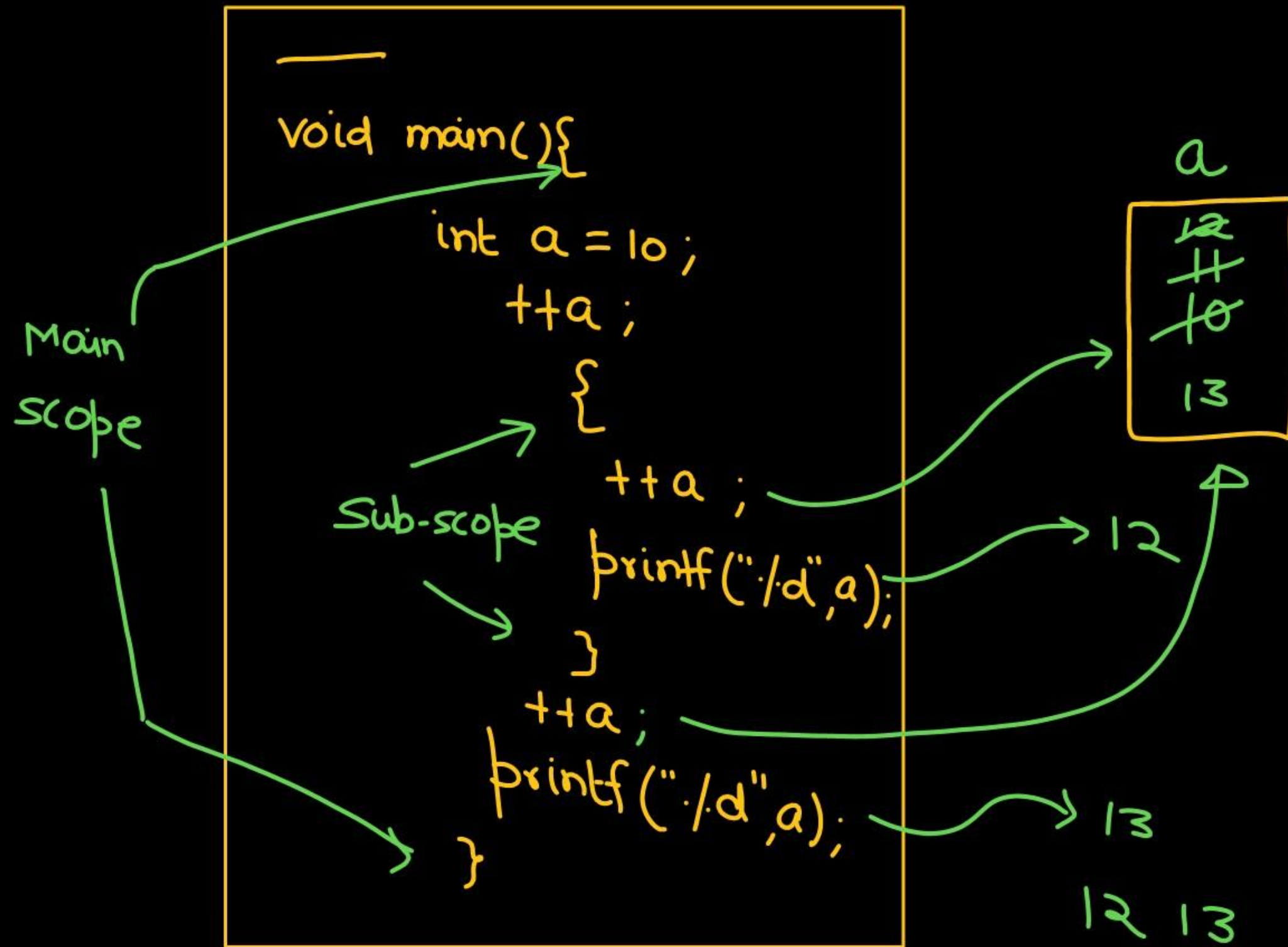
```
    printf("%d", a);
```

```
}
```

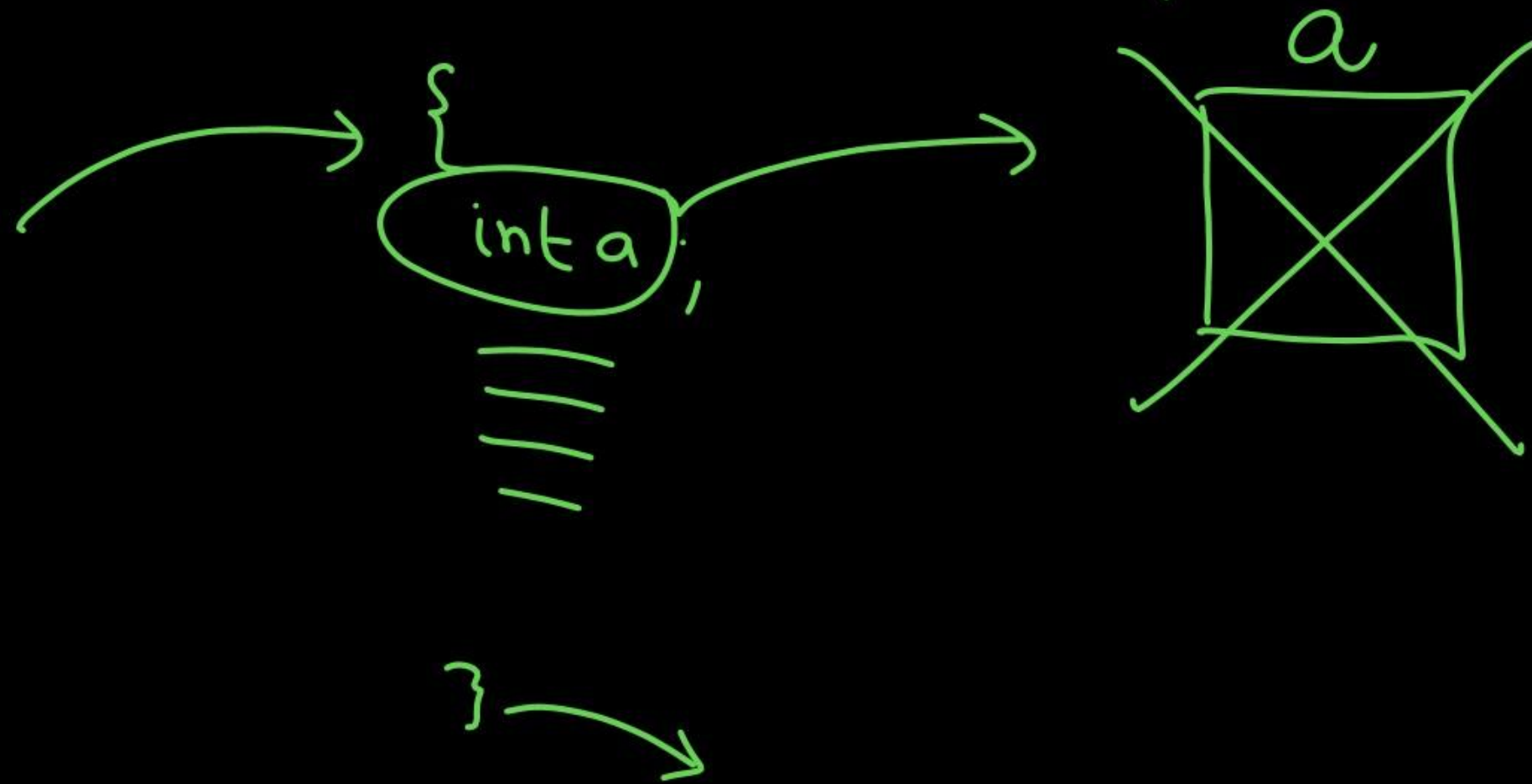
Garbage

```
void main(){  
    int a = 10;  
    ++a;  
    {  
        int a = 12;  
        ++a;  
        printf("/d", a);  
    }  
    ++a;  
    printf("/d", a);  
}
```





auto: created automatically when we enter the block in which they are declared and destructed automatically when u exit the block.



- ① for auto \Rightarrow scope and lifetime is same — block
- ② main-scope var. are accessible in sub-scope.
- ③ sub-scope var. are not accessible/available to main scope var.

12 18

```
void main() {  
    int a = 10;  
    ++a; {  
        ++a;  
        printf("/d", a);  
        {  
            int b = 5;  
            ++a;  
            printf("/d", a+b);  
            ++b;  
            printf("/d", b);  
            printf("/d", a);  
        }  
    }  
    ++a;  
    printf("/d", a);  
}
```

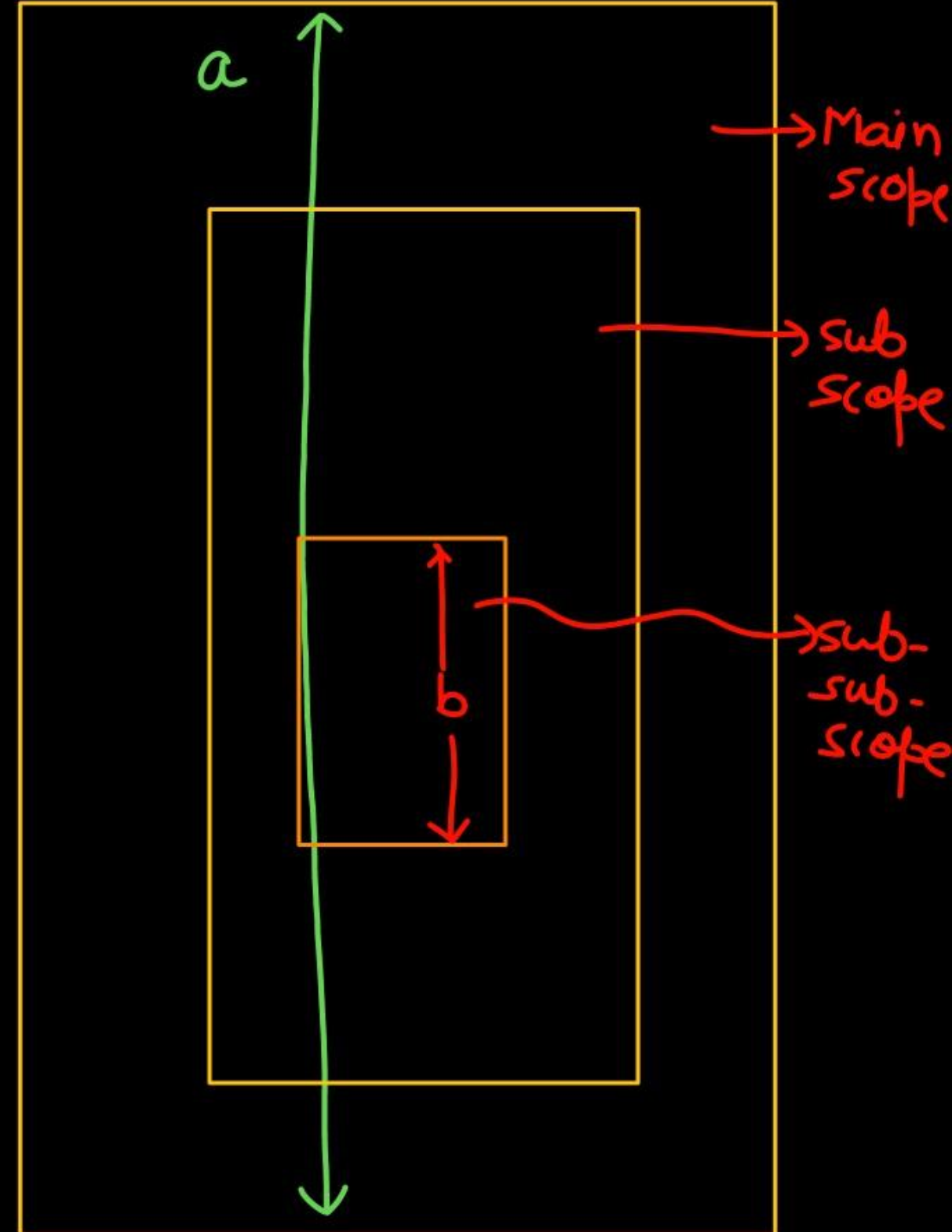


auto



Error

ud ke laal Padegi




```
void main(){
```

```
    int a = 0;
```

```
    ++a;
```

```
    {
```

```
        int a = 10;
```

```
        ++a;
```

```
        printf("/.d", a);
```

```
    }
```

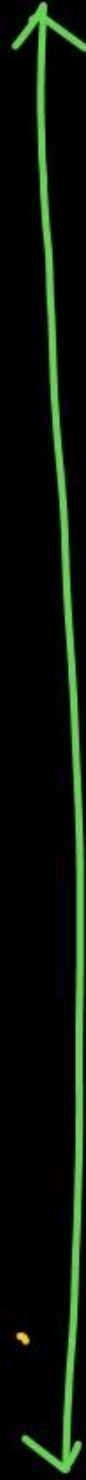
```
    ++a;
```

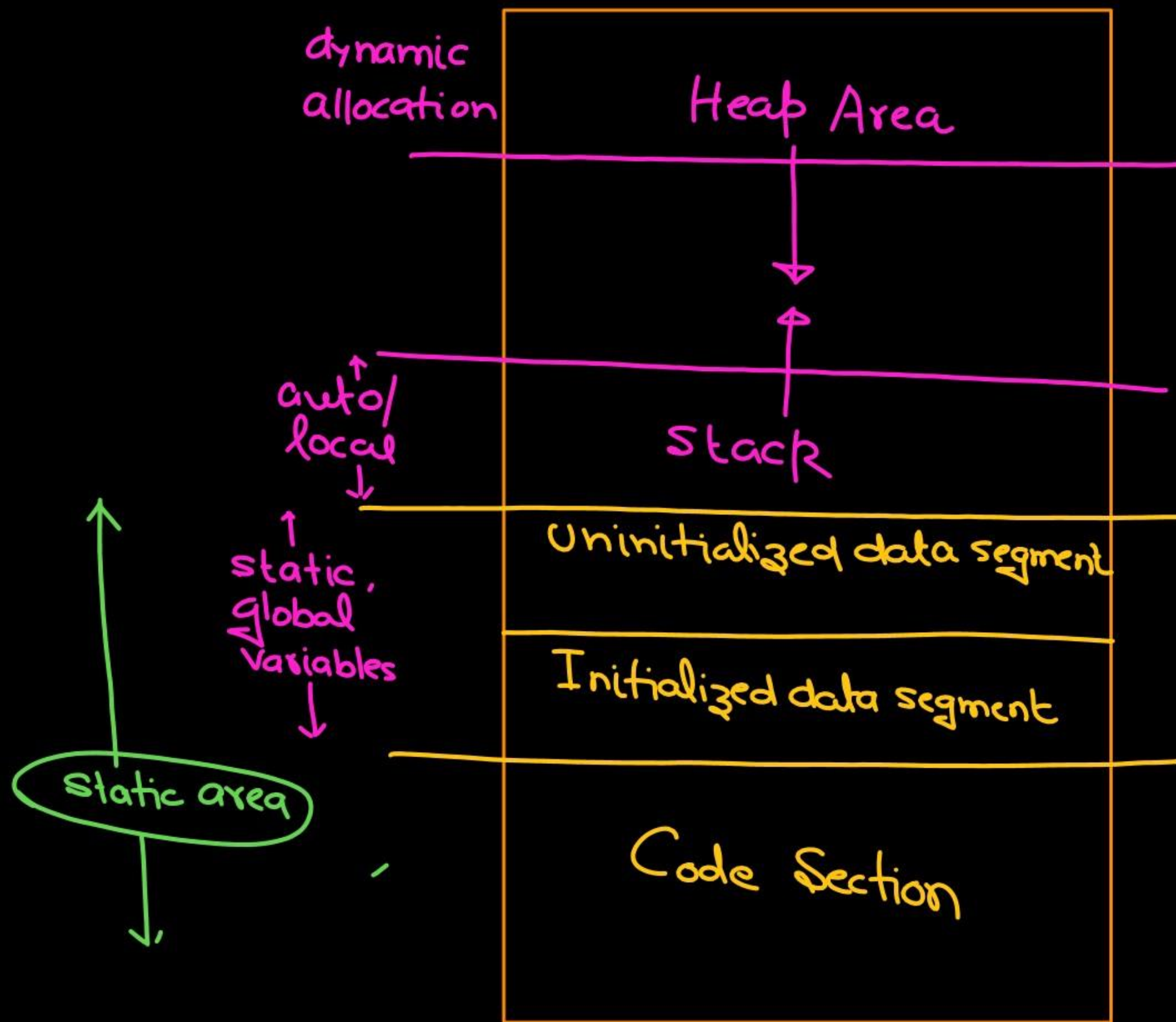
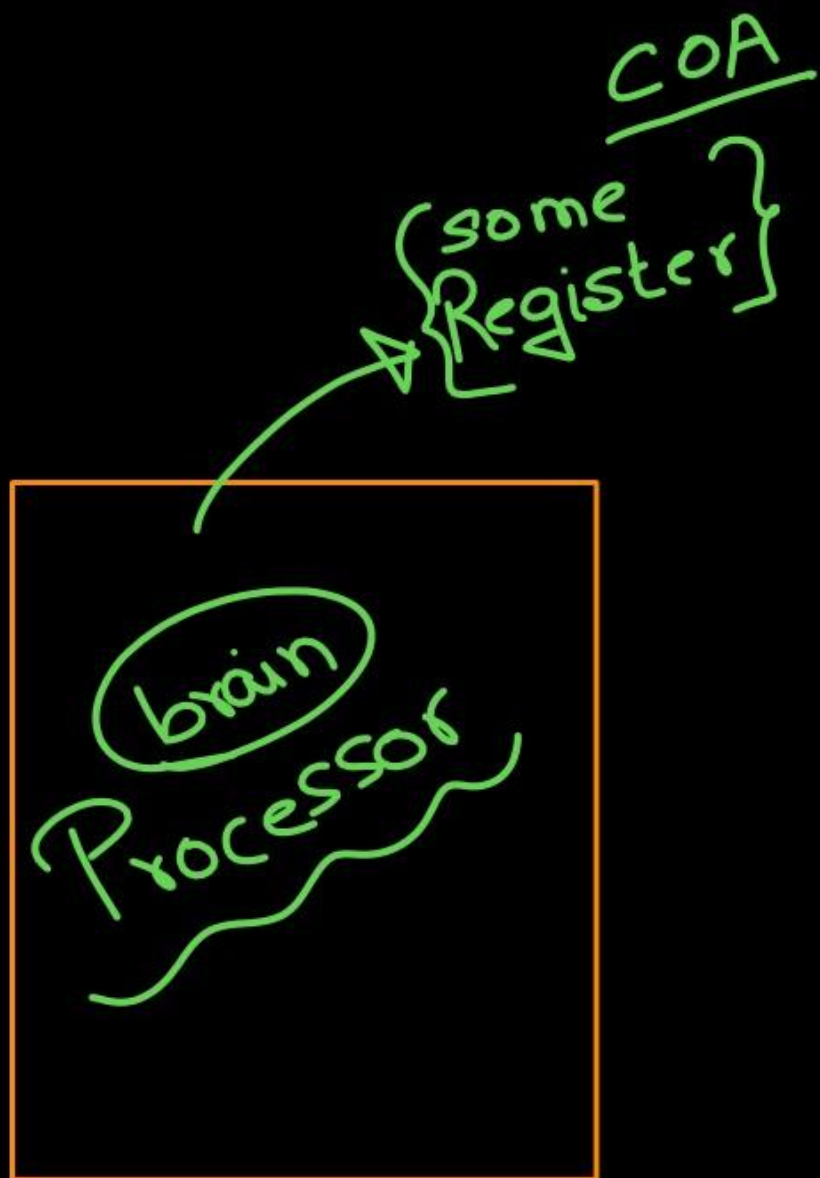
```
    printf("/.d", a);
```

```
}
```



11 2

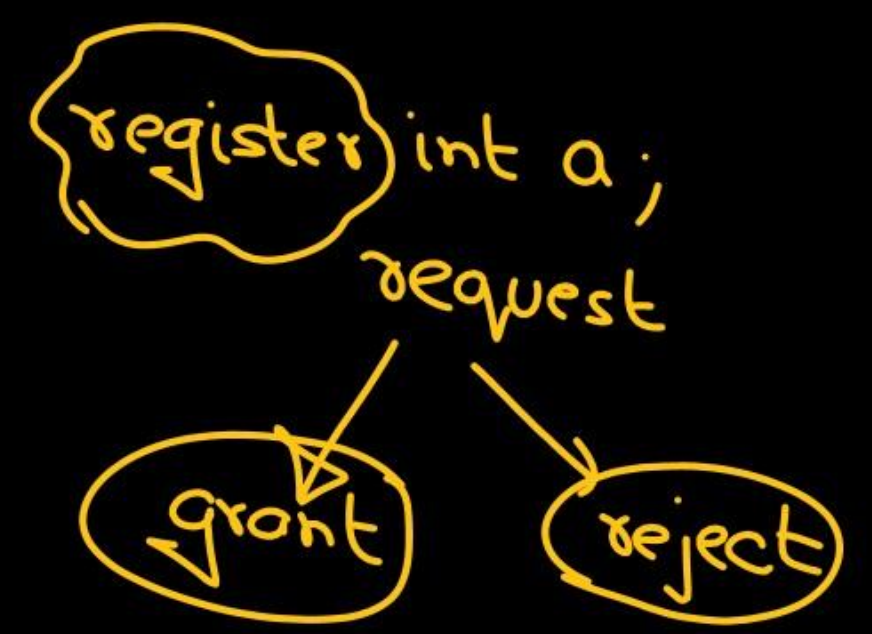




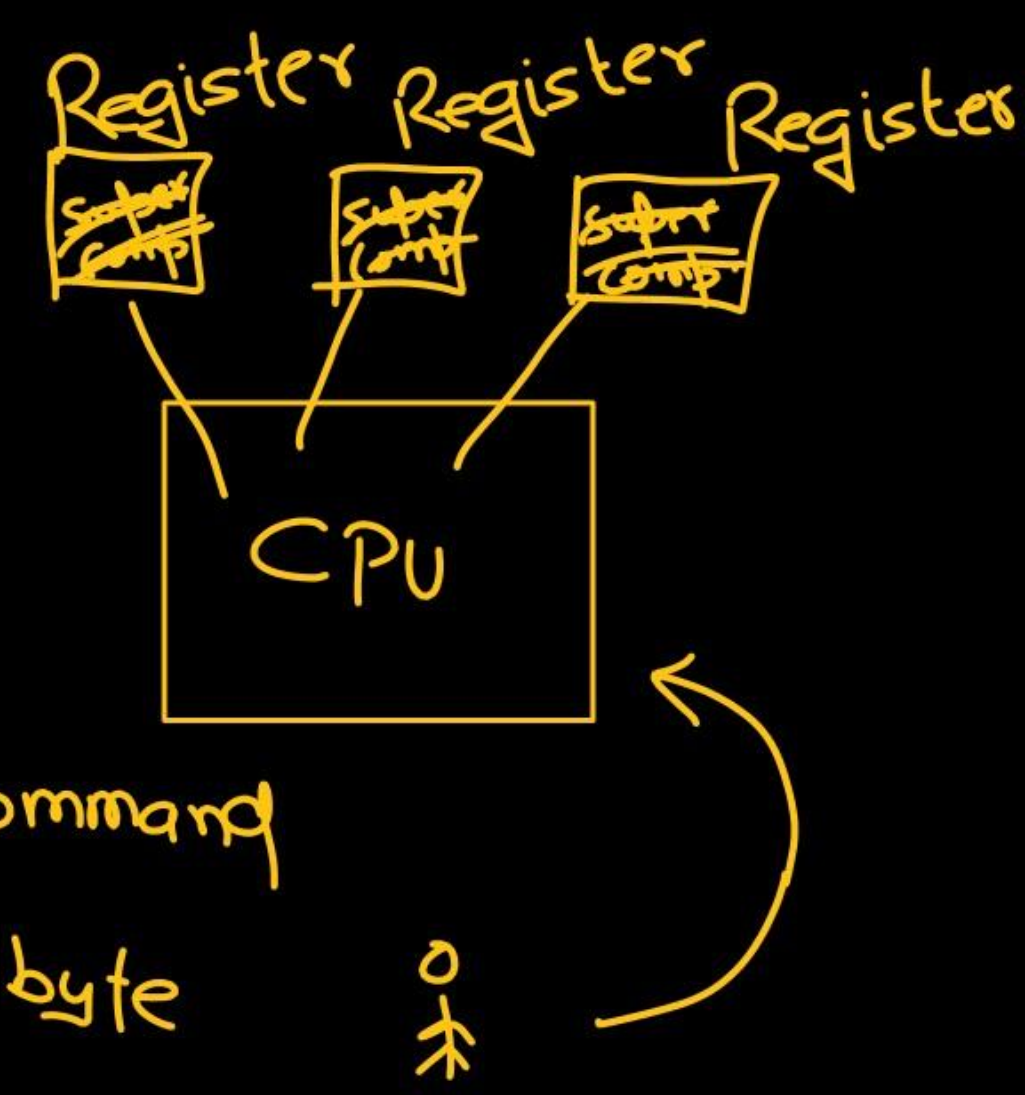
register

* As same as auto.

* storage : CPU register / stack



{ int a; }
3
⇒ Command
4 byte



{
register int a;
}

reject \Rightarrow {
int a; stack
}

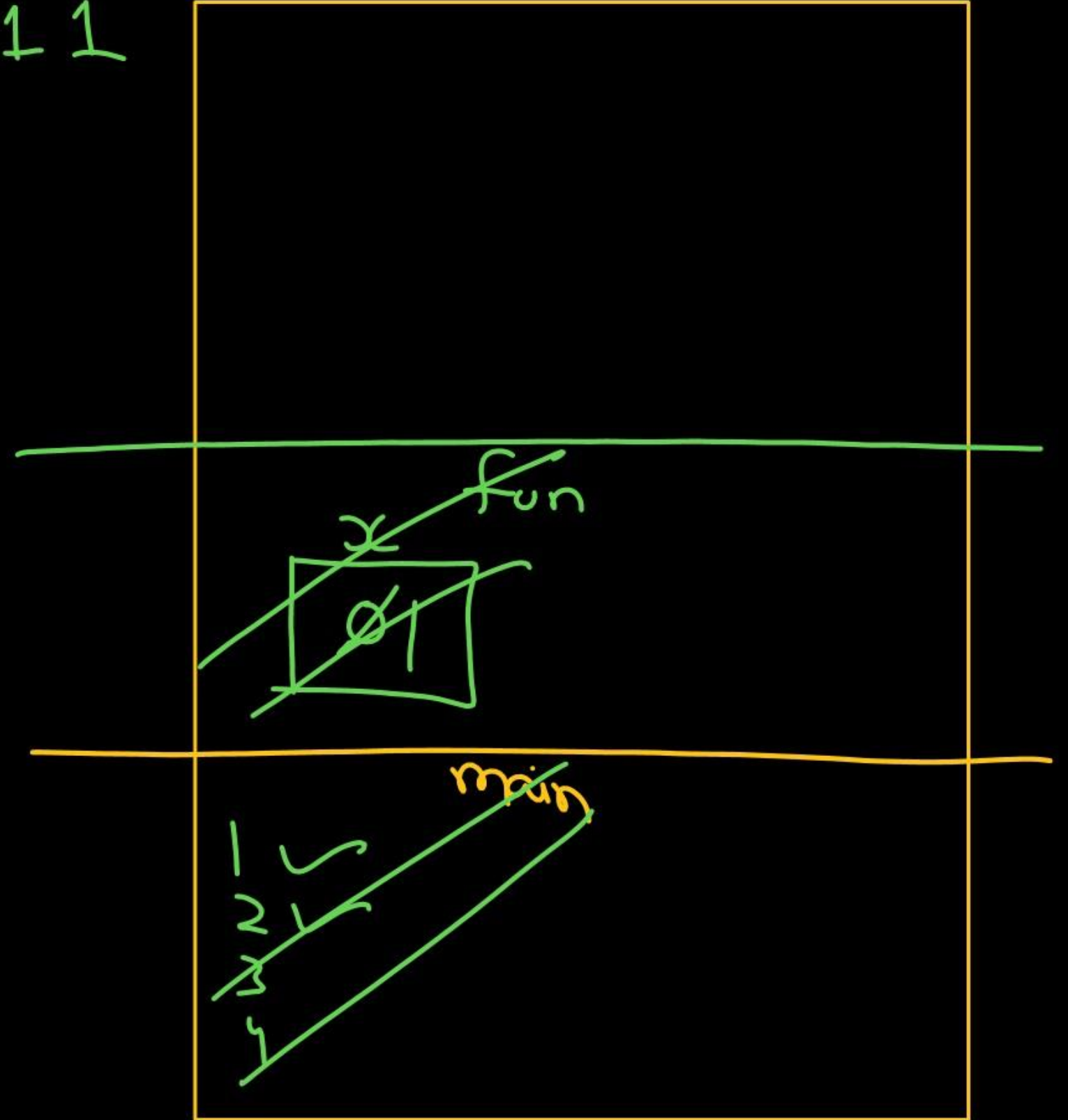
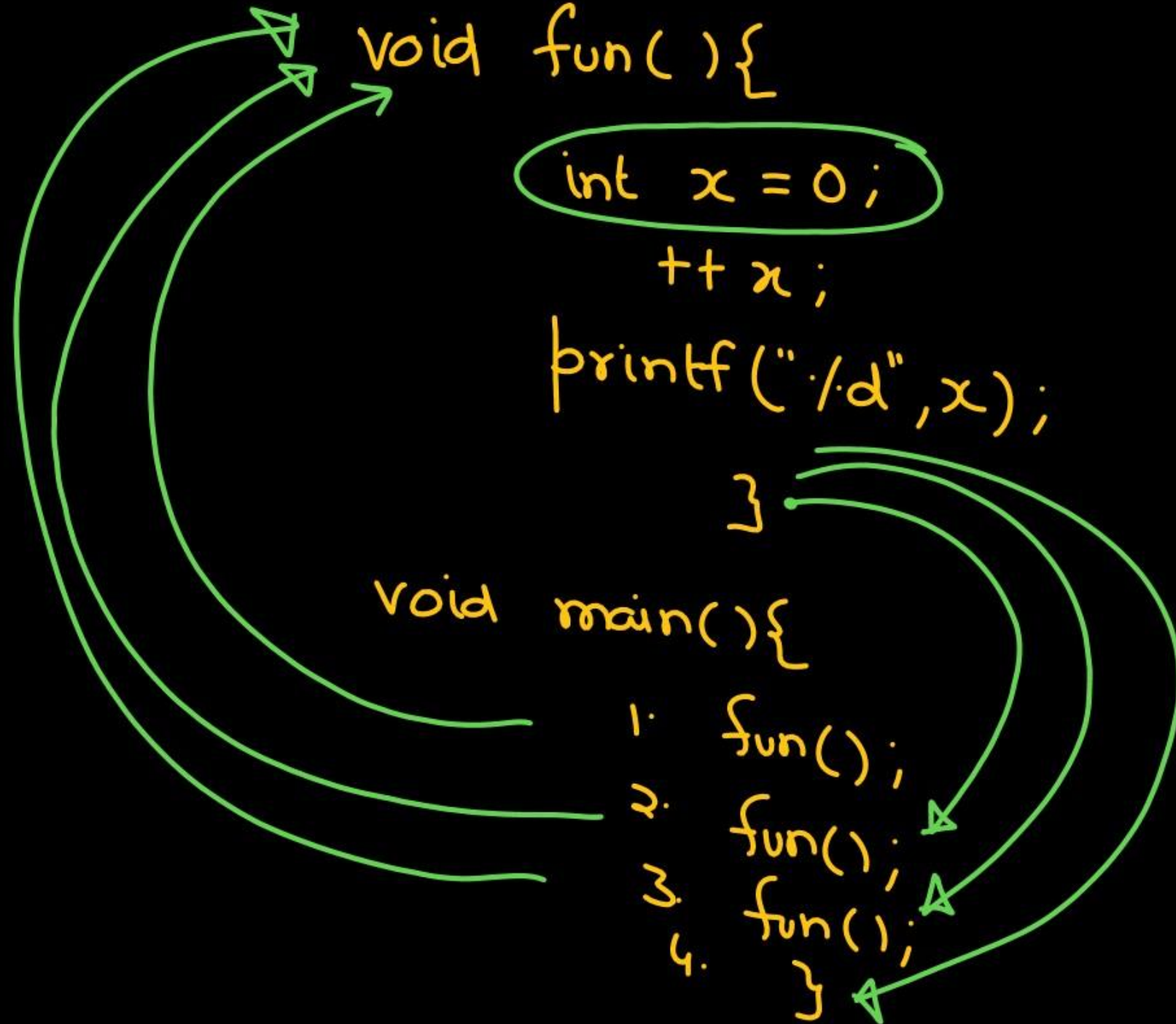
CPU register / stack

grant \Rightarrow {

CPU
register

}

111



Static variable

Lifetime : Program

scope : block

default : 0

Storage : static Area

① Value persist b/w diff function calls.

② No redeclaration

③ They are created only once in a program.

1 2 3

data segment

3
0 1 2

```
void fun(){
```

```
static int i = 0;
```

```
++i;
```

```
printf("/d", i);
```

```
}
```

```
void main(){
```

```
fun();
```

```
fun();
```

```
fun();
```

```
}
```

```

void fun() {
    int a = 0;
    static int b = 0;
    ++a;
    ++b;
    printf("%d %d", a, b);
}

```

Diagram illustrating variable scope in the function `fun()`:

- A vertical double-headed arrow labeled `b` spans the entire function body, indicating that `b` has a static scope.
- An arrow points from the opening curly brace of `fun()` to the line `int a = 0;`, indicating its local scope.
- A green oval highlights the line `static int b = 0;`.

scope

① `int a;`
`static int b = a;` X

```

void main() {
    int a = 10;
    ++a;
    ++b;
    printf("%d %d", a, b);
}

```

Diagram illustrating variable scope in the function `main()`:

- A green oval highlights the line `++b;`, indicating it refers to the static variable `b` defined in `fun()`.

`int a = 10, b = 0;`

```
void fun(){
```

```
    int a = 0;
```

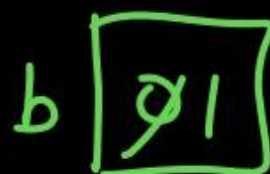
```
    static int b = 0;
```

```
    ++a;
```

```
    ++b;
```

```
    printf("/.d /.d", a, b);  
}
```

Scope



① int a;
static int b = a;

"/.d /.d"

↑
space

```
void main(){
```

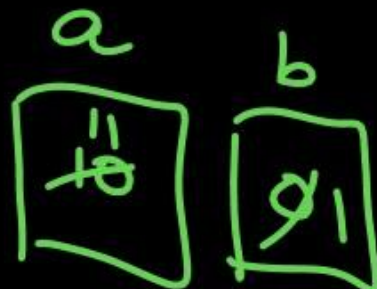
```
    fun();
```

```
    int a = 10, b = 0;
```

```
    ++a;
```

```
    ++b;
```

```
    printf("/.d /.d", a, b);  
}
```



local to main

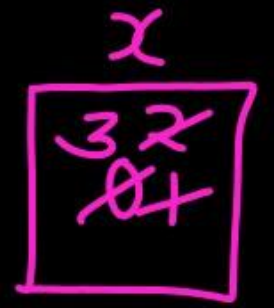
global → variable outside all the function

```
int x ;  
void f1( ) {  
  
}  
void f2( ) {  
  
}  
  
void main( ) {  
  
}
```

Compilation
Top to
bottom

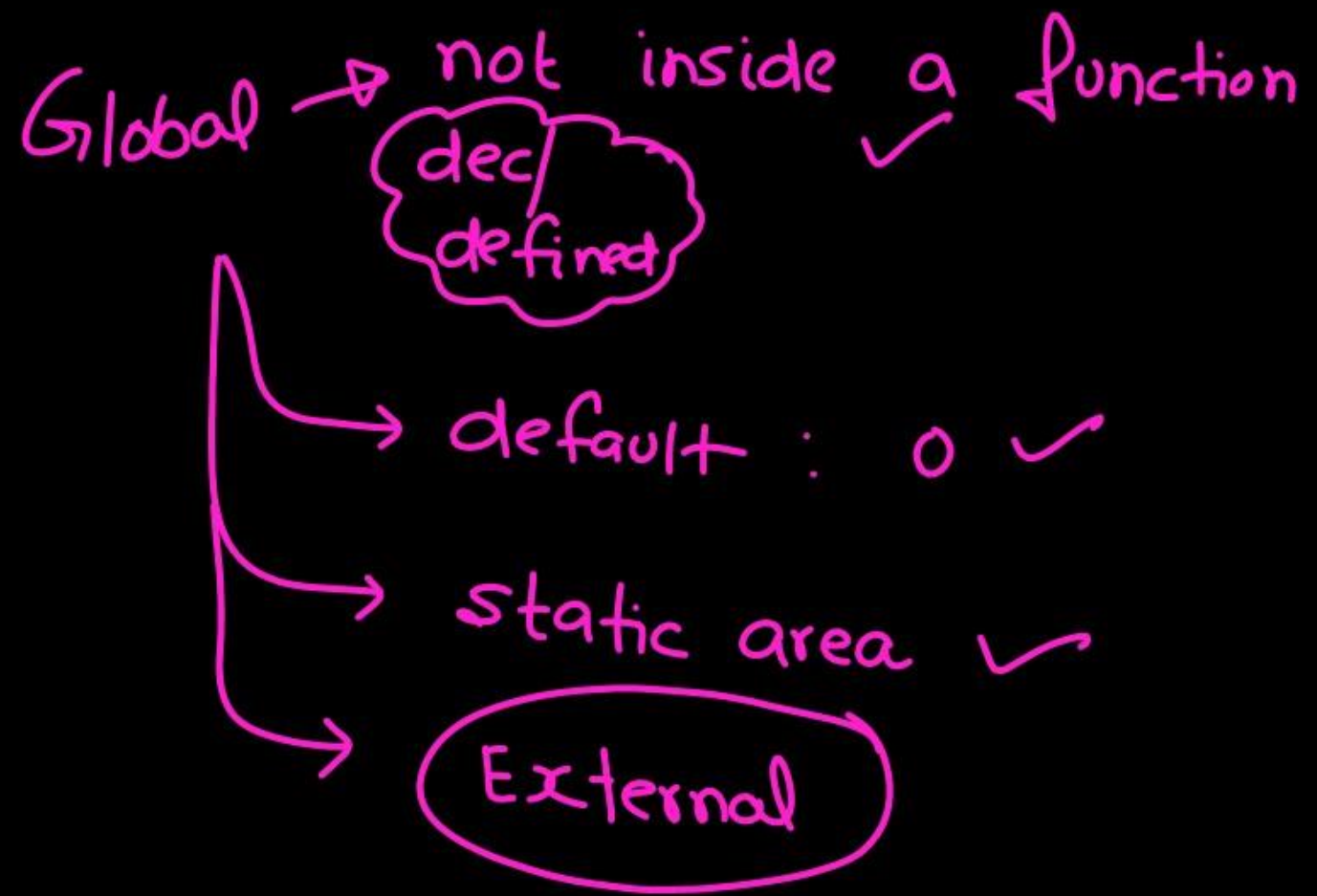
```
int x;
void f1(){
    ++x;
    printf("%d", x);
}
```

```
void f2(){
    ++x;
}
```



13

```
void main(){ f1(); f2();
             ++x;
             printf("%d", x);
}
```



void f1(){
 ++x;
 printf("/.d", x);
}

↑
x2

→ Error

int x = 10;

void f2(){
 ++x;
 printf("/.d", x);
}

void main(){
 f1();
 f2();
 ++x;
 printf("/.d", x);
}

```
void f1() {  
    extern int x;  
    ++x;  
    printf("/.d", x);  
}
```

forward decl

```
int x = 10;
```

```
void f2() {  
    ++x;  
    printf("/.d", x);  
}
```

```
void main() {  
    f1();  
    f2();  
    ++x;  
    printf("/.d", x);  
}
```

No Error

```
void f1() { extern int x; (No Error  
++x;  
}
```

```
void f2() { extern int x; (No memory)  
++x;  
}
```

int x = 10;

```
void main() {  
++x;  
f1();  
f2();  
printf("%d", x);  
}
```



```
void f1() {  
    extern int x;  
    ++x;  
}
```

```
void f2() {  
    extern int x;  
    ++x;  
}
```

```
void main() {  
    extern int x;  
    ++x;  
    f1();  
    f2();  
    printf("/d", x);  
}
```

```
int x = 10;
```

local


✓

extern int x;

```
void f1() {  
    ++x;  
}
```

```
void f2() {  
    ++x;  
}
```

```
void main() {  
    ++x;  
    f1();  
    f2();  
    printf("/d", x);  
}
```

An illustration on a dark blue background with faint circuit patterns. A woman with long dark hair, wearing a white long-sleeved shirt and dark blue pants, is sitting on a stack of four books. She is using a laptop. To her right is a large computer monitor with an orange base and a light blue frame. The monitor's screen is white and displays the text 'THANK YOU GW SOLDIERS !' in bold black capital letters. Above the monitor is a stylized globe with blue lines. To the left of the monitor is a white Wi-Fi symbol. Below the monitor are two white paper clips. A white paper airplane is flying above the globe.

**THANK YOU GW
SOLDIERS !**