CS & IT ENGINEERING

Programming in C

Functions and Storage Classes
Lec- 02

By- Pankaj Sharma sir

TOPICS TO BE COVERED

- Storage Classes

```
for( i=1; i<=n; i=i*3)
    {
    for( j=i; j<=n; j++)
        {
        pf ___
        }
    }
```

$i=1$	$i=3^1$	$i=3^2$	$\cdots$	$i=3^k$
$j=1$ to n	$j=3^1$ to n	$j=3^2$ to n		$j=3^k$ to n
$(n-1+1)$	$n-3^1+1$	$n-3^2+1$		$n-3^k+1$

$$(n-1+1)+(n-3^1+1)+(n-3^2+1)+\cdots(n-3^k+1)$$

$$(n+1)(k+1)-1-3^1-3^2-\cdots-3^k$$

$$=(n+1)(k+1)-(1+3+3^2+\cdots 3^k)$$

$$(n+1)(k+1)-\frac{(3^{k+1}-1)}{3-1}$$

$$3^{k} <= n$$

$$k <= \log_3 n$$

$$k = \lfloor \log_3 n \rfloor$$

$$= \boxed{(n+1)(\lfloor \log_3 n \rfloor + 1) - \frac{\left(3^{\lfloor \log_3 n \rfloor + 1} - 1\right)}{2}}$$

Storage class

1.) scope : part of program/code in which a variable can be accessed. (visibility of variable).

2.) Lifetime : Duration (Active/Alive)

3.) Default value : What is the value of a variable if we don't initialize it.

4.) Storage Area : Where a variable is stored.

```
#include<stdio.h>
void main(){
    int a;
    printf("%d",a);
}
```

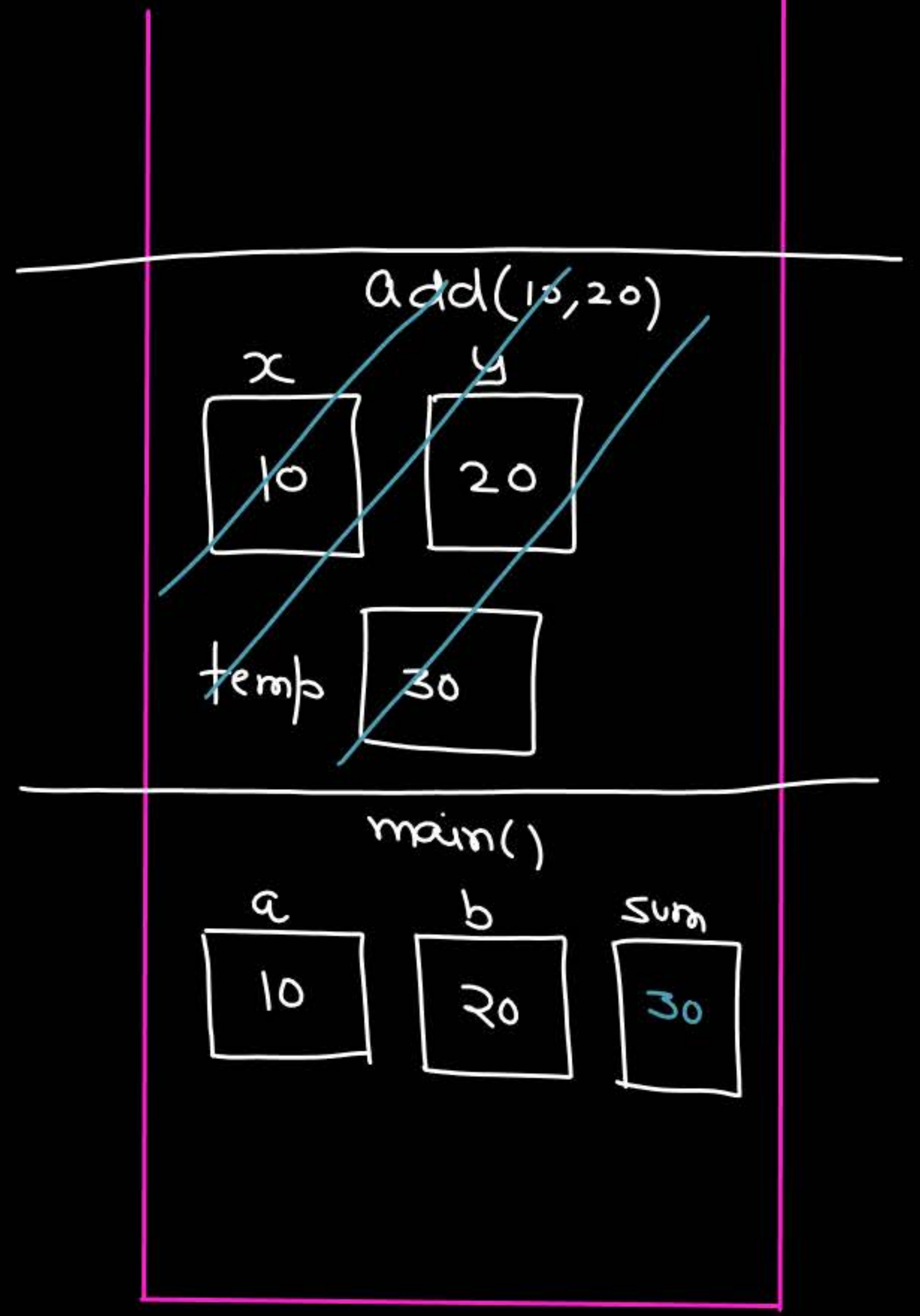
No initialization

auto

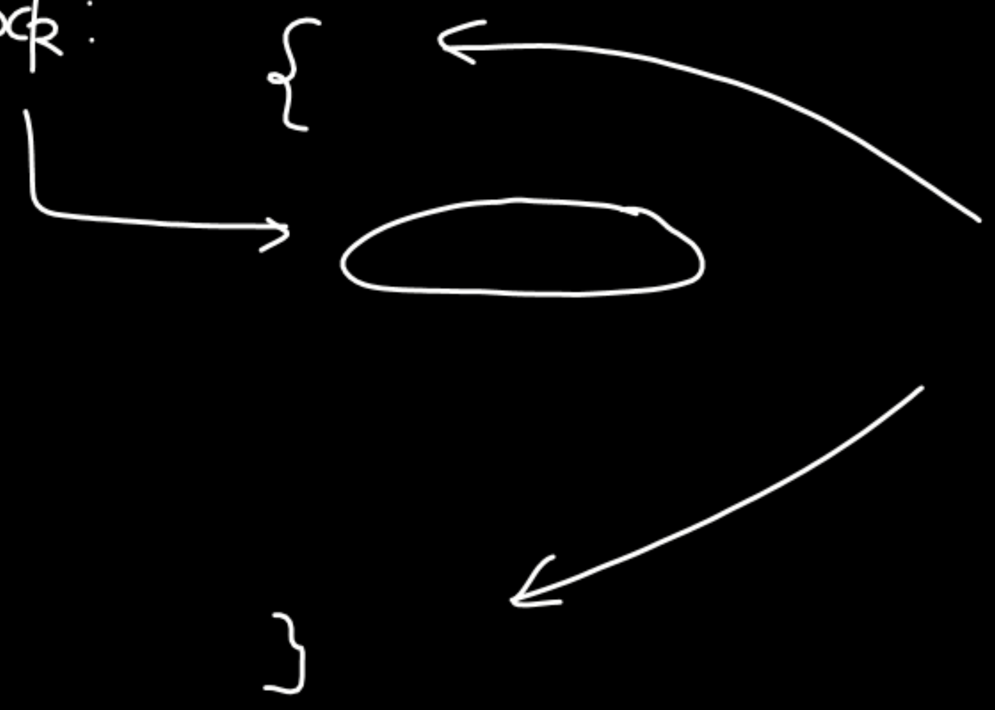
```c
#include<stdio.h>
int add(int,int);

void main(){

    int a=10,b=20,sum;
    sum= add(a,b);
    printf("%d",sum);
}
```

formal arguments

```c
int add(int x,int y)
  {
     int temp;
     temp = x+y;
     return temp;
  }
```



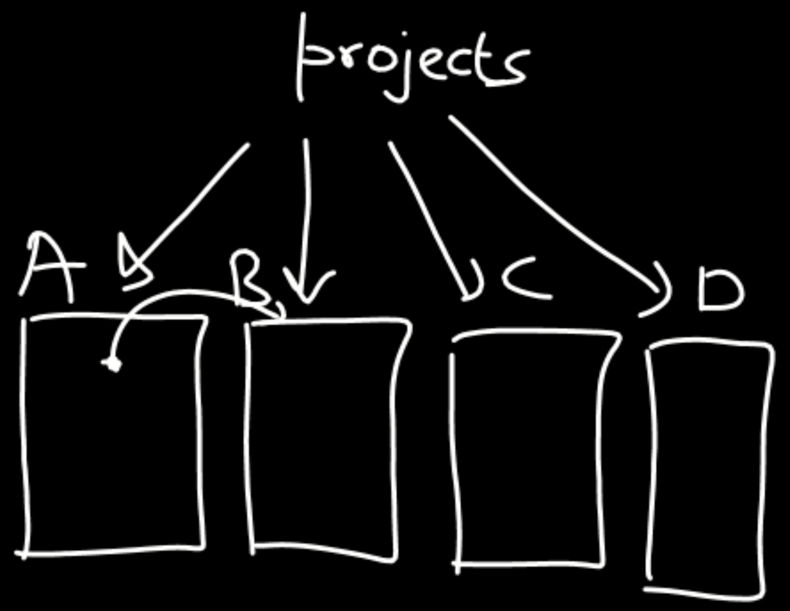
1.) block:



2.) File

```
void f1(){
          int a;
       =
       =
       }
void f2(){
        int z;
          =
          =
          }
void main(){
          int y=10;
             =
             =
             }
```

3. Multiple files



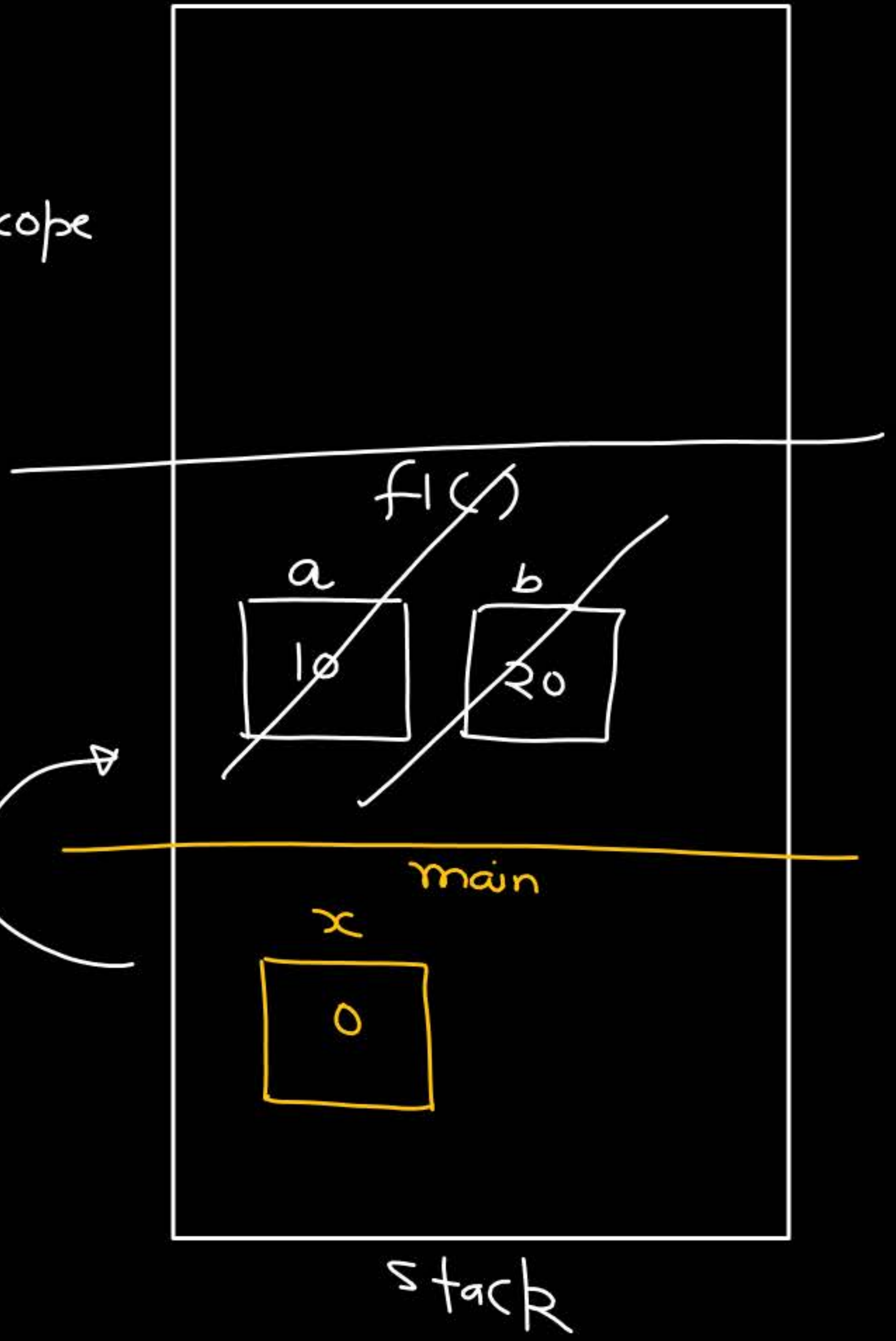
```
void f1(){
    int a=10,b=20;
     printf("%d",a+b);
    }
void main(){
      int x=0; f1();
      printf("%d",a+x);
      }
```

a का scope

?

Error

O/P
30



1.) by default, variables declared inside a function are auto variables.

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

OR

```
void f1(){

   auto int a;
            =
          }
```

auto

1.) Scope : block in which they are declared.

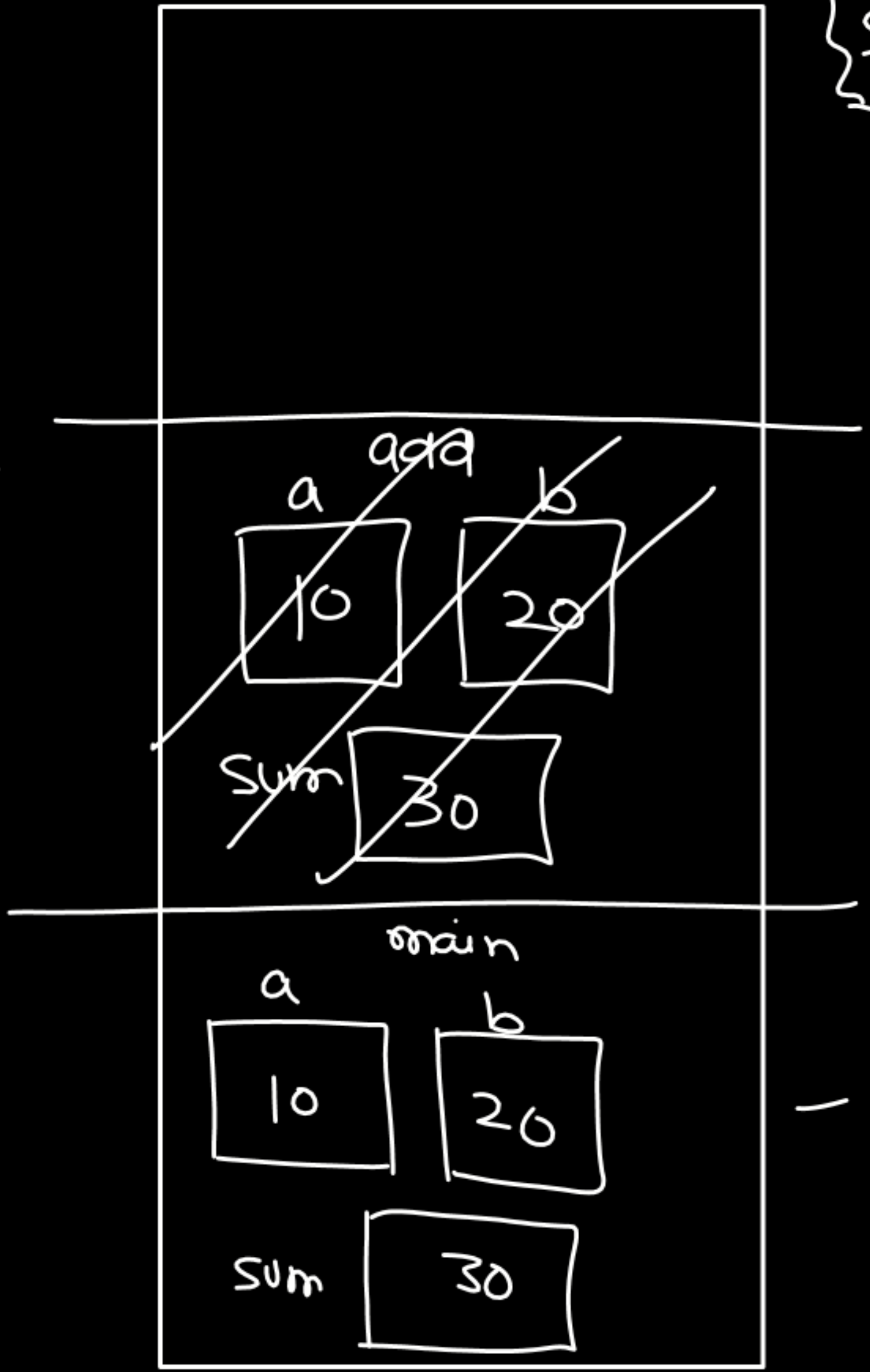
2.) Lifetime: block in which they are declared.

3.) Default value : Garbage

4.) Storage Area : Stack

scope

```
void main(){

    int a=10,b=20,sum;

    sum = add(a,b);

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



```
int add(int a, int b)
    {
     int sum;
     sum=a+b;
     return sum;
    }
```

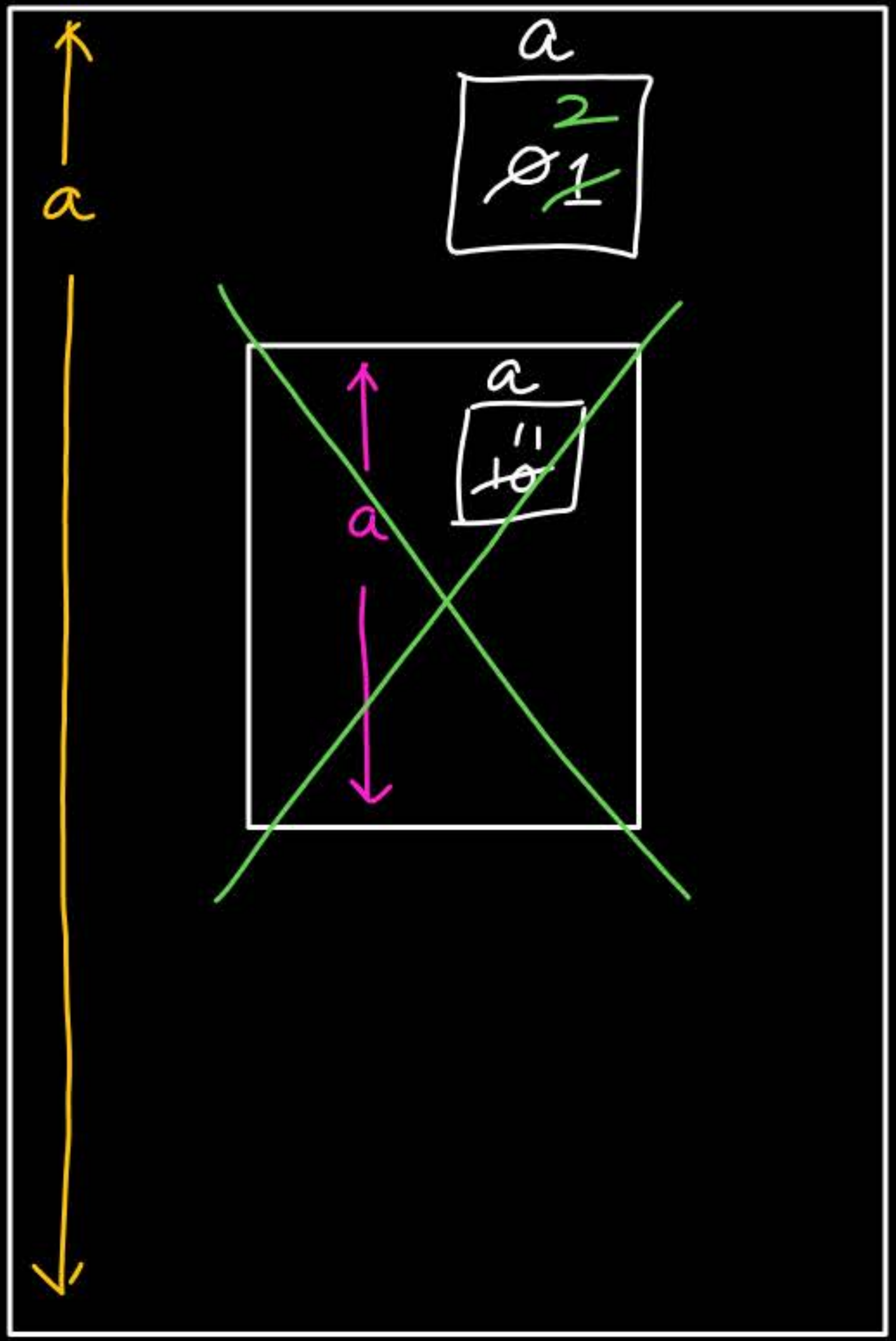
```
void main(){
      int a = 0; ✓
         ++a;
         {
          int a = 10; ✓
            ++a;
          printf("%d",a);   → 11
         }
        ++a;
        printf("%d",a);

        }
```

new scope

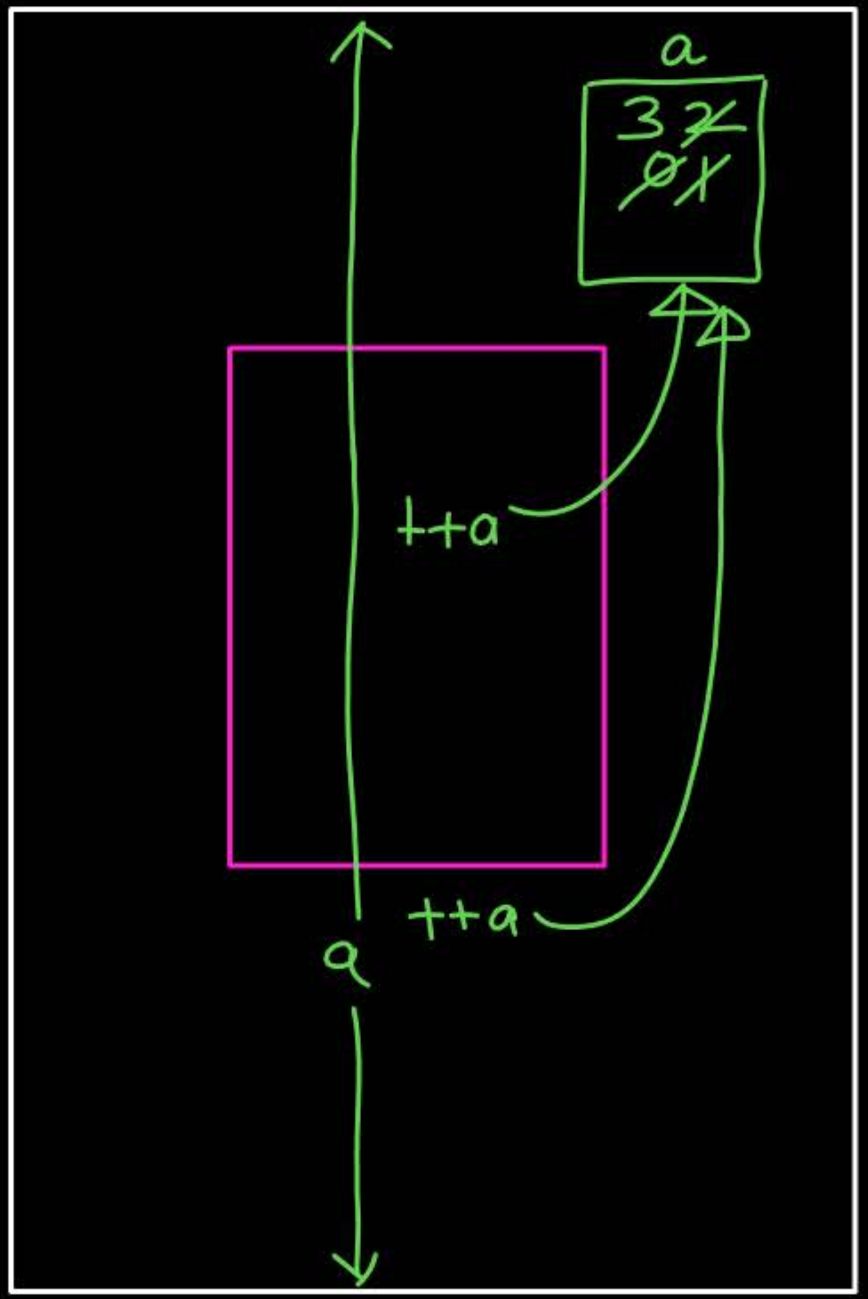
sub-scope

MAIN Scope

O/P : 112



```
void main(){
      int a = 0; ✓
        ++a;
         {

            ++a;
          printf("%d",a);     → 2
         }
        ++a;
        printf("%d",a);      → 3

         }
```



```
void main(){
      int a = 0;  ✓
         ++a;
         {

            ++a;
          printf("%d",a);
         }
         ++a;
         printf("%d",a);

         }
```

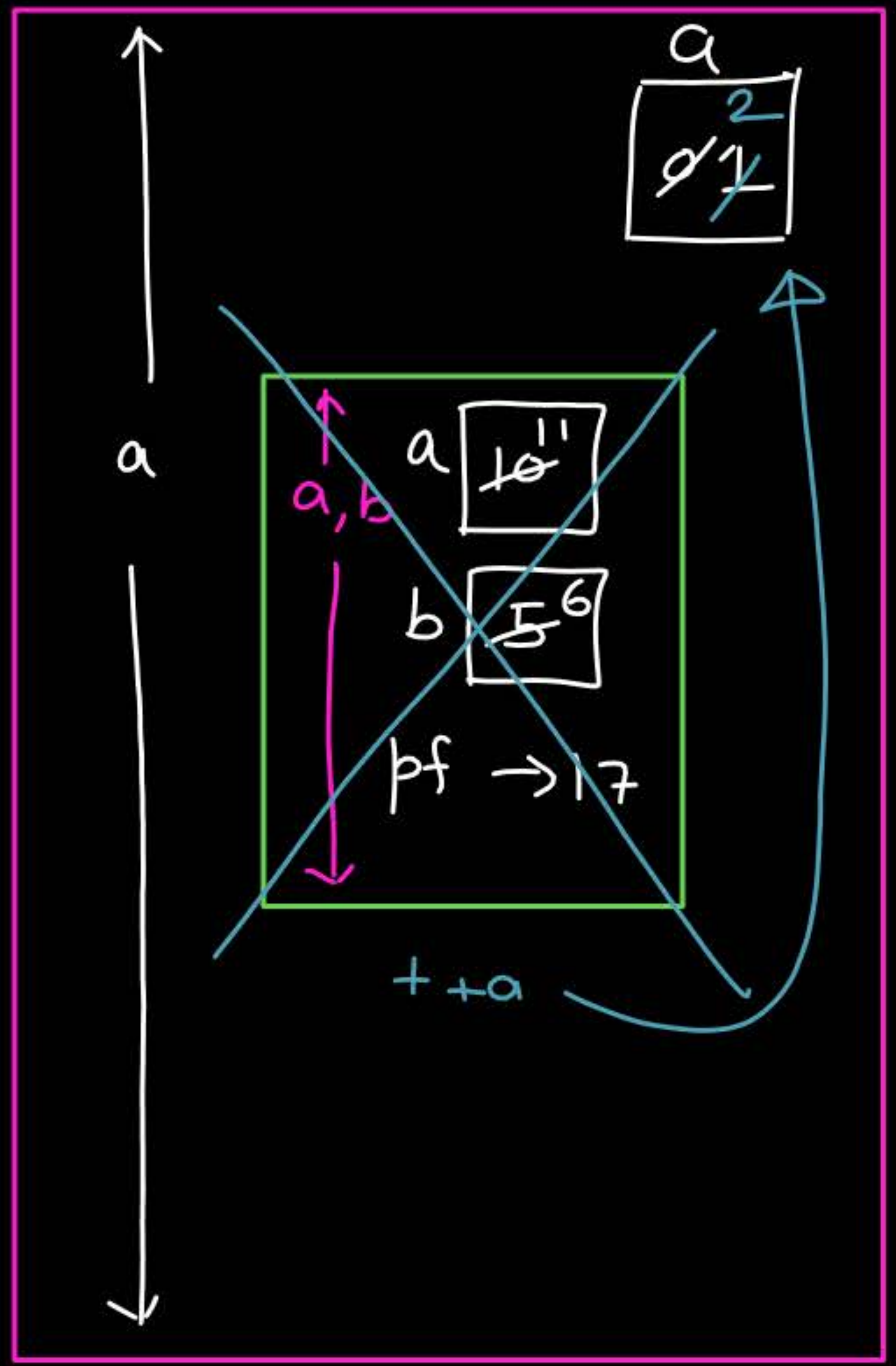
within main → 1 scope

function

block

```
void main(){
    int a = 0; ✓
        ++a;
         {
         int a= 10,b= 5; ✓
          ++a; ++b;
          printf("%d",a+b);   → 17
         }
         ++a;
        printf("%d",a+b);    → b? Error
          }
```

← b is a variable in sub-scope

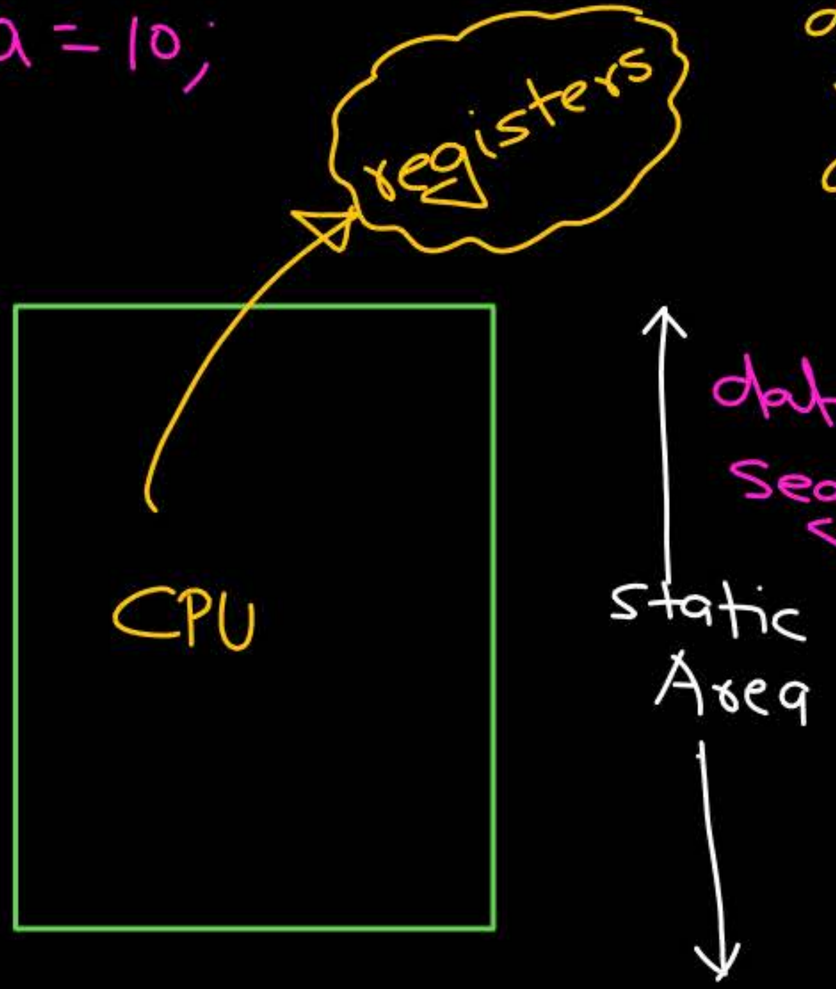


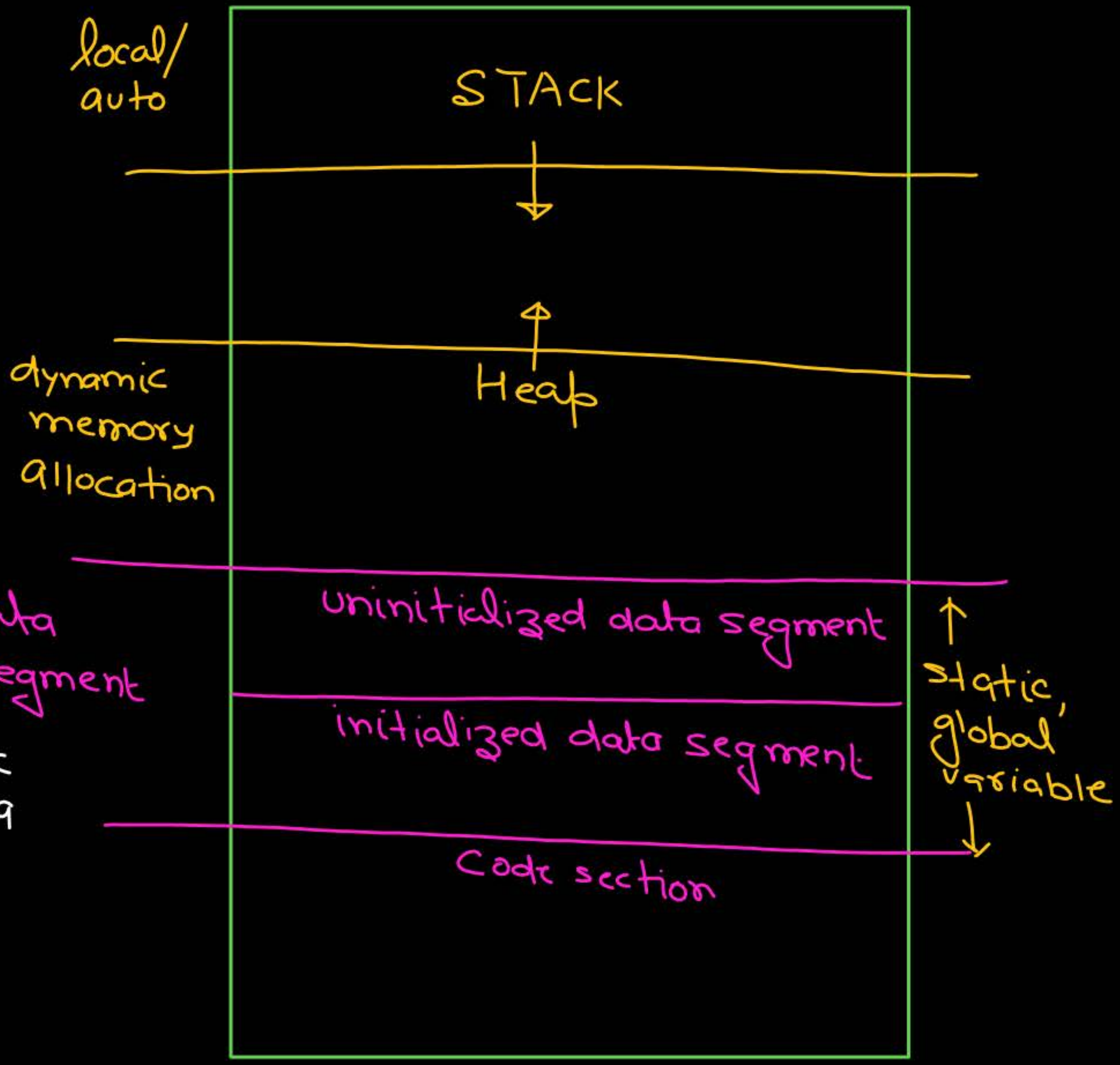
1) Main scope variables are accessible to sub-scope variables.

2) sub-scope variables are not accessible to main scope variables.

3) Local/auto variables are created $\boxed{\text{automatically}}$ when we enter the block in which they are declared and destructed $\boxed{\text{automatically}}$ when we exit the block.

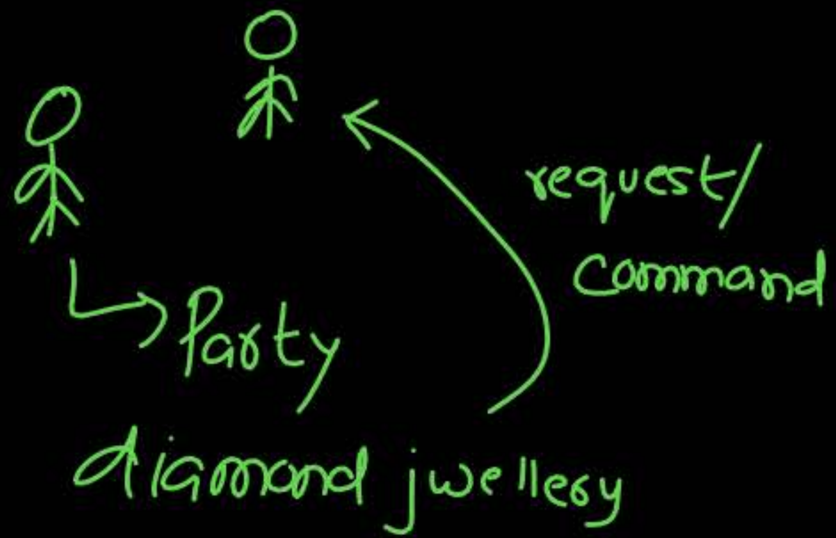
```
int a;

int a=10;
```

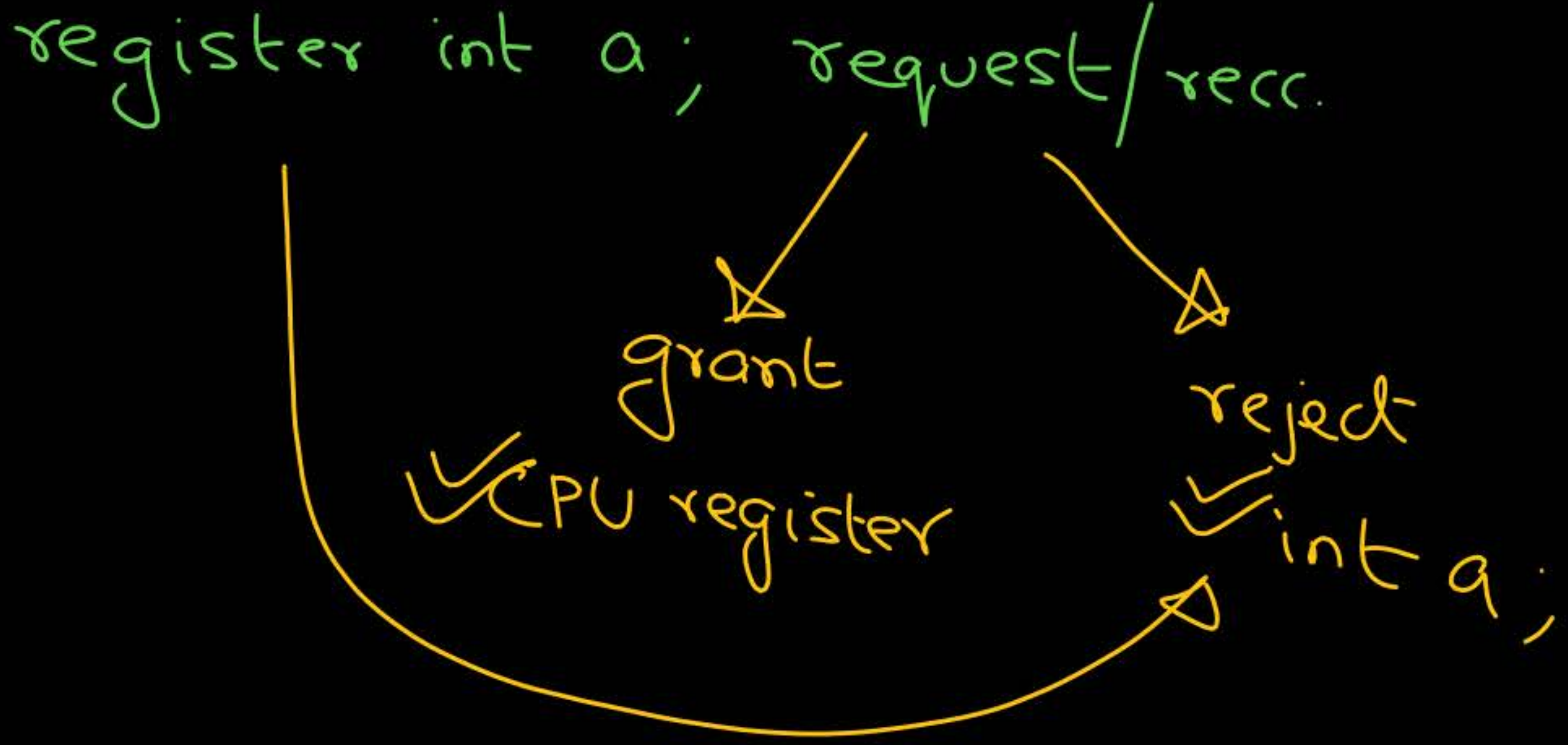

registers
CPU
data Segment
static Area


local/ auto
STACK
Heap
dynamic memory allocation
uninitialized data segment
initialized data segment
static, global variable
Code section

register

* As same as auto
* storage Area : $\boxed{\text{CPU register/stack}}$ ✓



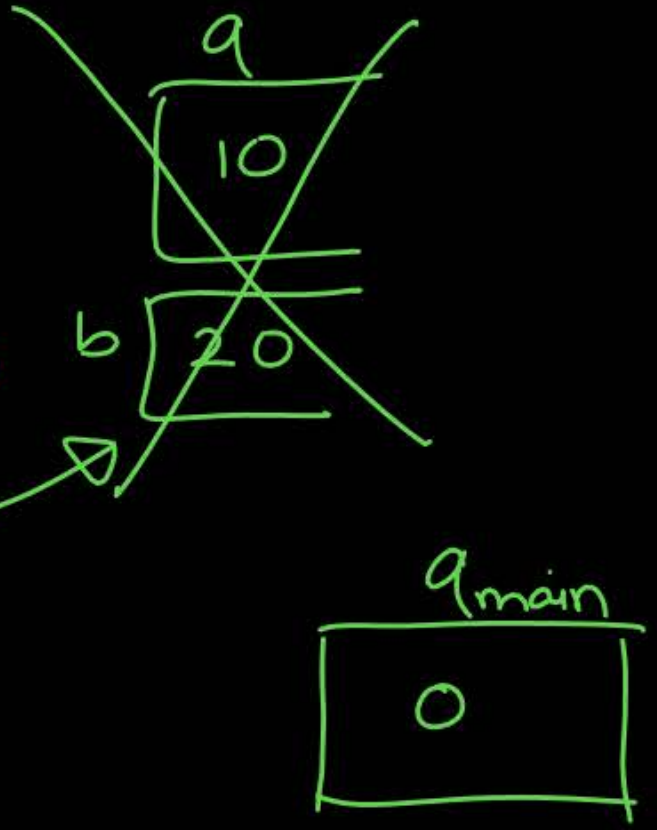


```
register int a ; ✗
scanf("%d", &a);
```

→ OK if register is not granted

→ Error if register is granted — address of

```
void fun( ){
    int a=10,b=20;
    printf("%d",a+b);
}              30
void main(){
    int a=0;
    fun();
    printf("%d",b); ⇒ Error
}
```

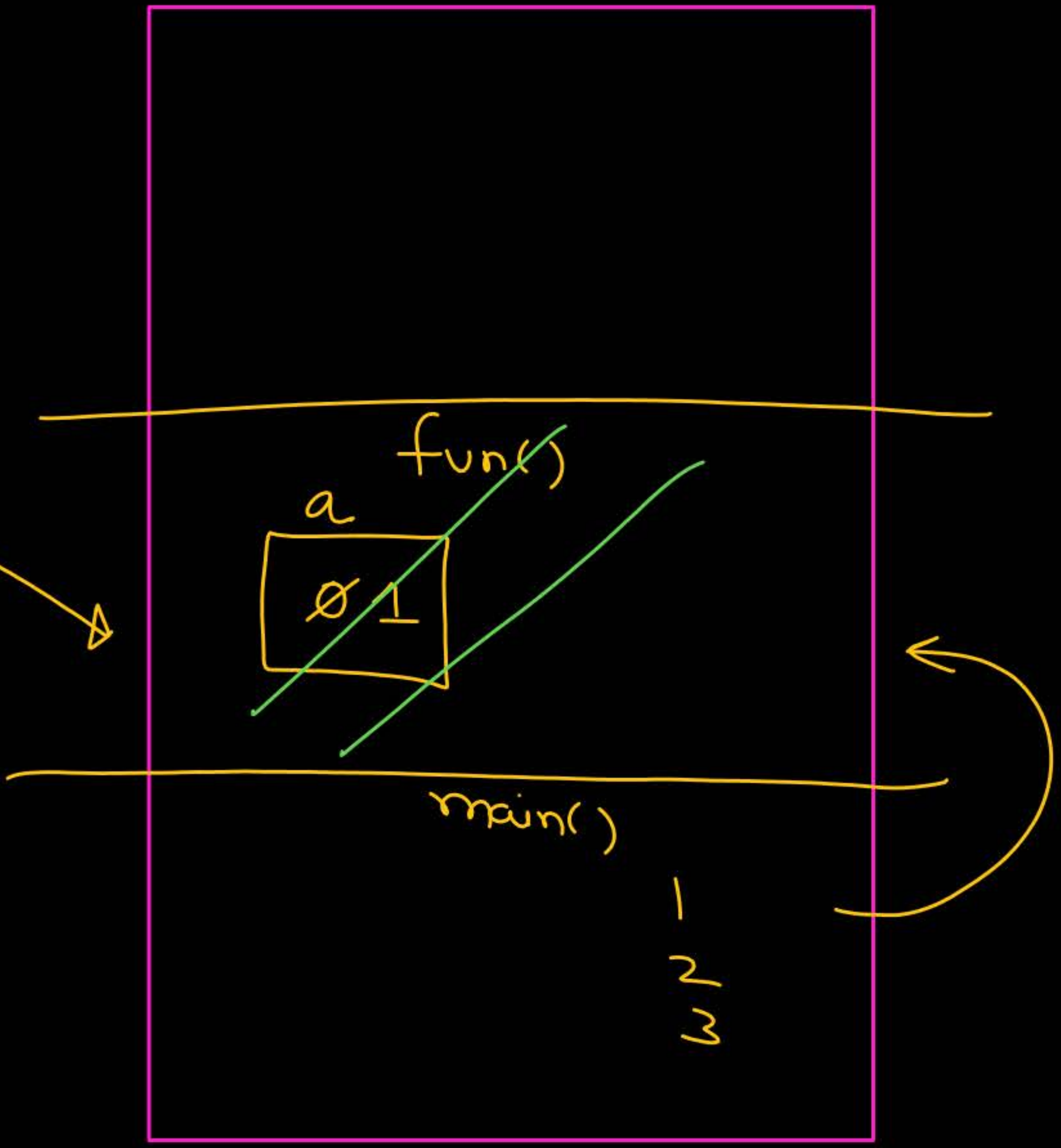


local

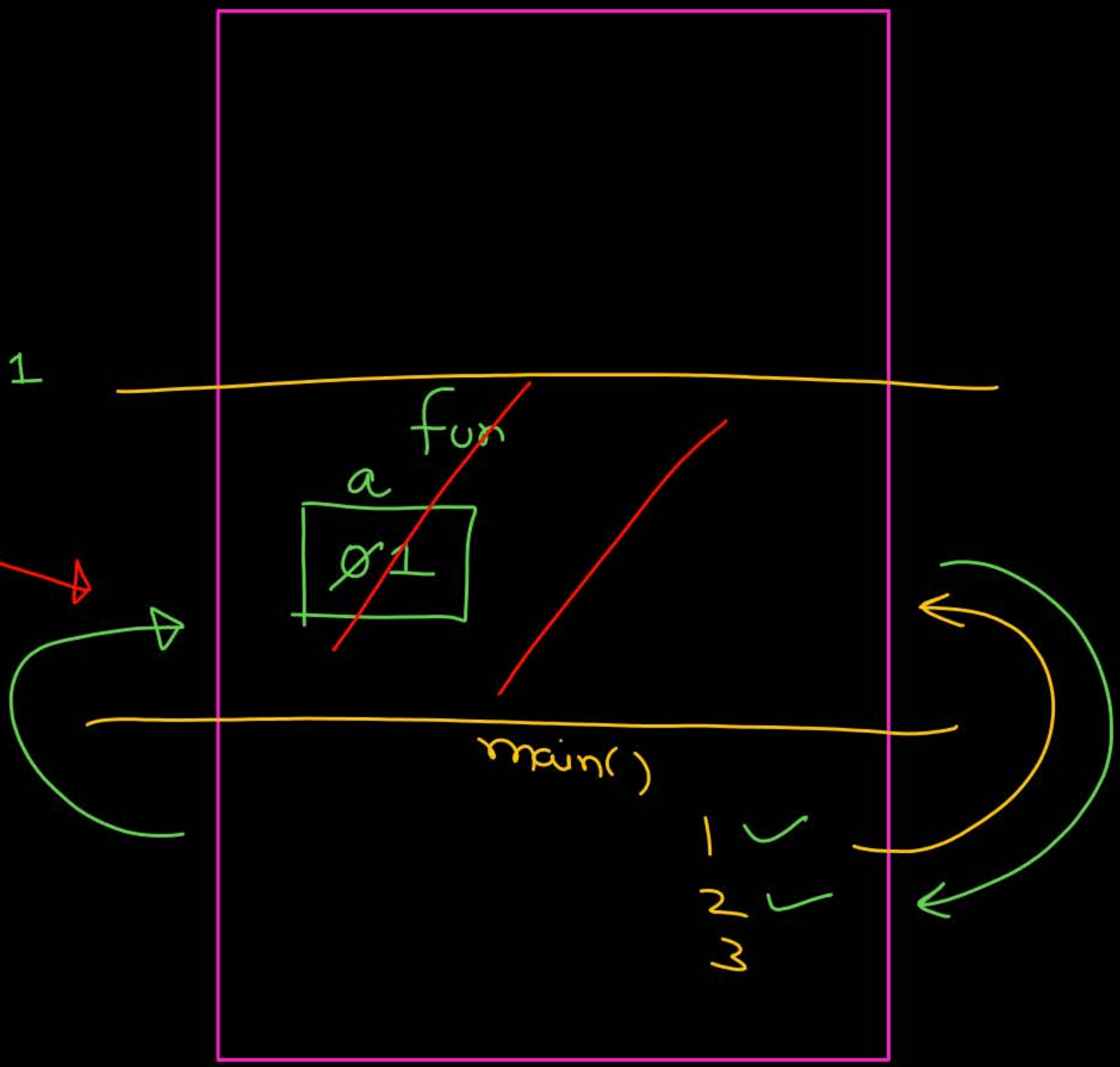
scope, lifetime

→ same

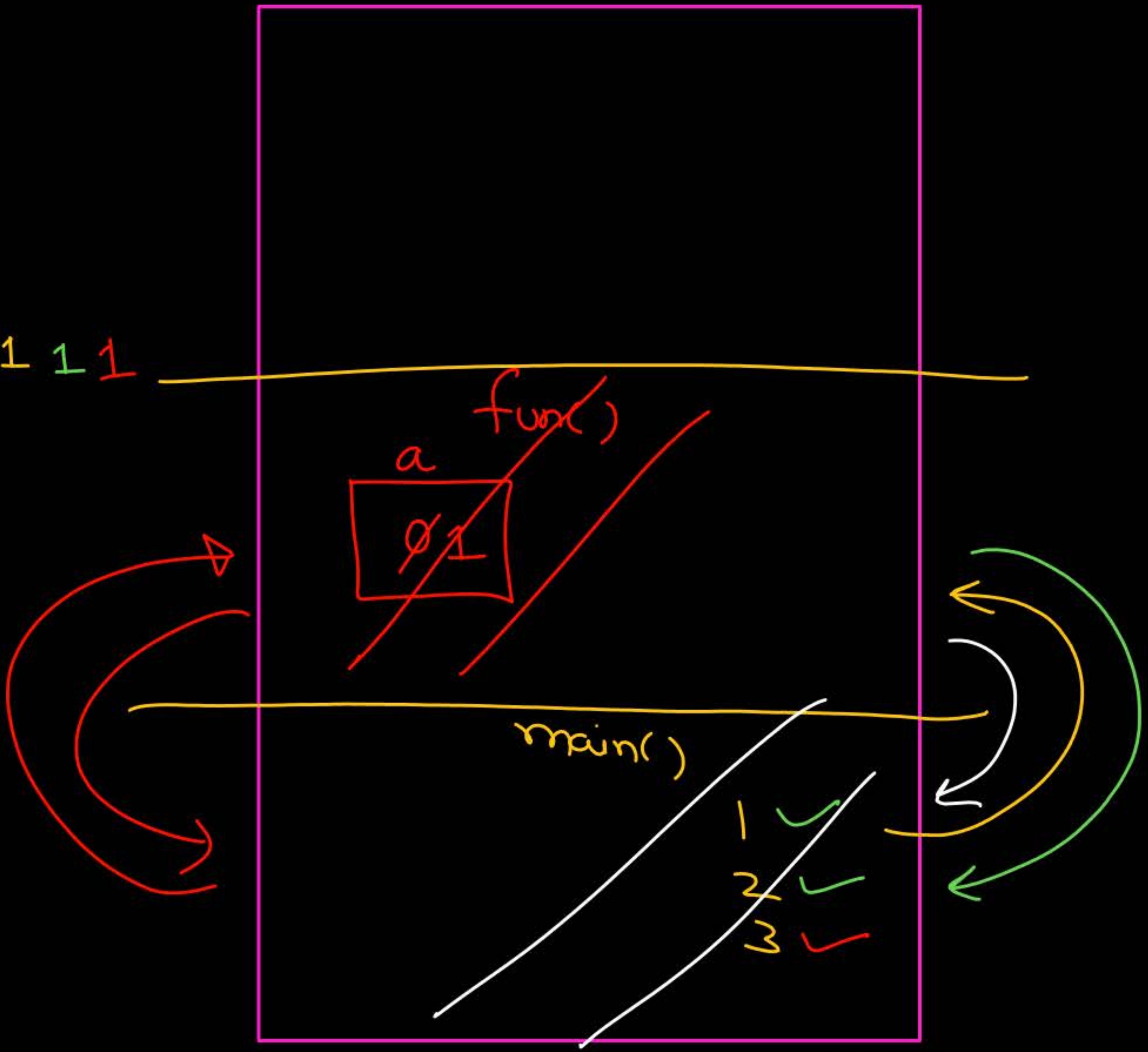
```
void fun(){
    int a = 0;
     ++a;
    printf("%d",a);  → 1
  }

void main(){
  1  fun();
  2  fun();
  3  fun();
   }
```



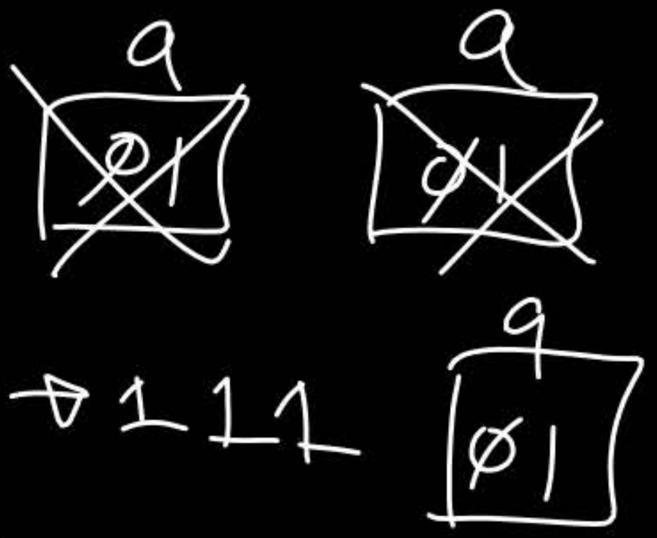
```
void   fun(){
   int a = 0;
     ++a;
   printf("%d",a); → 1 1

     }

void main(){

  1  fun();
  2  fun();
  3  fun();
     }
```



```
void   fun(){
   int a = 0;
     ++a;
   printf("%d",a);  → 1 1 1
     }

void main(){
   1  fun();
   2  fun();
   3  fun();
      }
```



```
void   fun(){
 int a = 0;
  ++a;
 printf("%d",a);  → 1 1 1
  }

void main(){
 1  fun();
 2  fun();
 3  fun();
  }
```



static

scope : block in which they are declared.

Lifetime : Program

Default : 0

Storage Area : static area (data segment)

1. The value persist between different function calls.
2. No redeclarations.
3. They are created only once in the program

```
void fun(){
    ~~static int i~~ ;
        ++i ;
    printf("%d",i);
        }

void main(){
        fun();
        fun();
        fun();
            }
```

→ Activation record ✗

→ data segment

→ 1 2 3

i

$\boxed{\cancel{0}\ \cancel{1}\ \cancel{2}\ 3}$

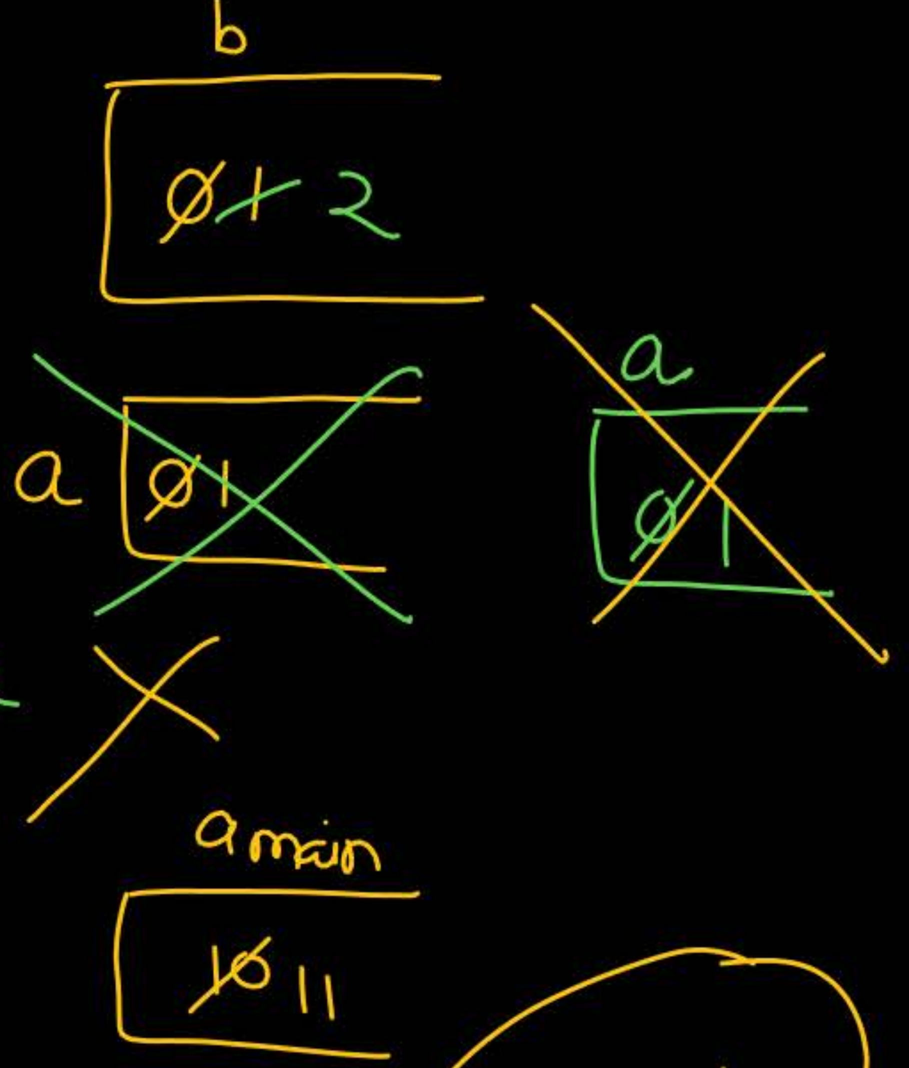
```
void f(){
    int a = 0;
    ~~static int b = 0;~~
    ++a;
    ++b;
    printf("%d %d", a, b);
}

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

scope of b

11 12

What is the scope of b ?

Error



global variables → by default variable defined outside all function

```
void f1(){

}

void f2(){

}
```

```
void main(){

}
```

Top to bottom

Compilation

```
int x = 0; ✓
void f1(){
        ++x; ✓
        }

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

global variables → by default variable defined outside all function

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

int x = 10;
void f2(){
        ++x;        ✓
    }
void main(){
        ++x;
        f1();
        f2();       ✓
    printf("%d",x);
    }
```

To avoid C.E

⇒ forward declaration

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

int x = 10;
void f2(){
        ++x;
    }
void main(){
        ++x;
        f1();
        f2();
        printf("%d",x);
    }
```

(info.)

To avoid C.E

$\Rightarrow$ forward declaration

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

void f2(){
    extern int x;
    ++x;
}
int x = 10;
void main(){
    ++x;
    f1();
    f2();
    printf("%d",x);
}
```

define

local forward declaration
(global variable)

```c
extern int x;    // global forward declaration
void f1(){

        ++x;
      }

void f2(){

          --x;
        }
int x = 10;
void main(){
        ++x;
        f1();
        f2();
      printf("%d",x);
      }
```

(global variable)

No memory is Created (info)

define

Memory is created

declaration/define $\Longrightarrow$ for global variable these are different

Physical memory is created

(i) scope :

(ii) Lifetime

(iii) 0

(iv) static Area (data segment)

THANK YOU GW SOLDIERS !