CS & IT ENGINEERING

C Programming

Function & Storage Classes

Lec- 04

By- Pankaj Sharma Sir

Storage Classes 02

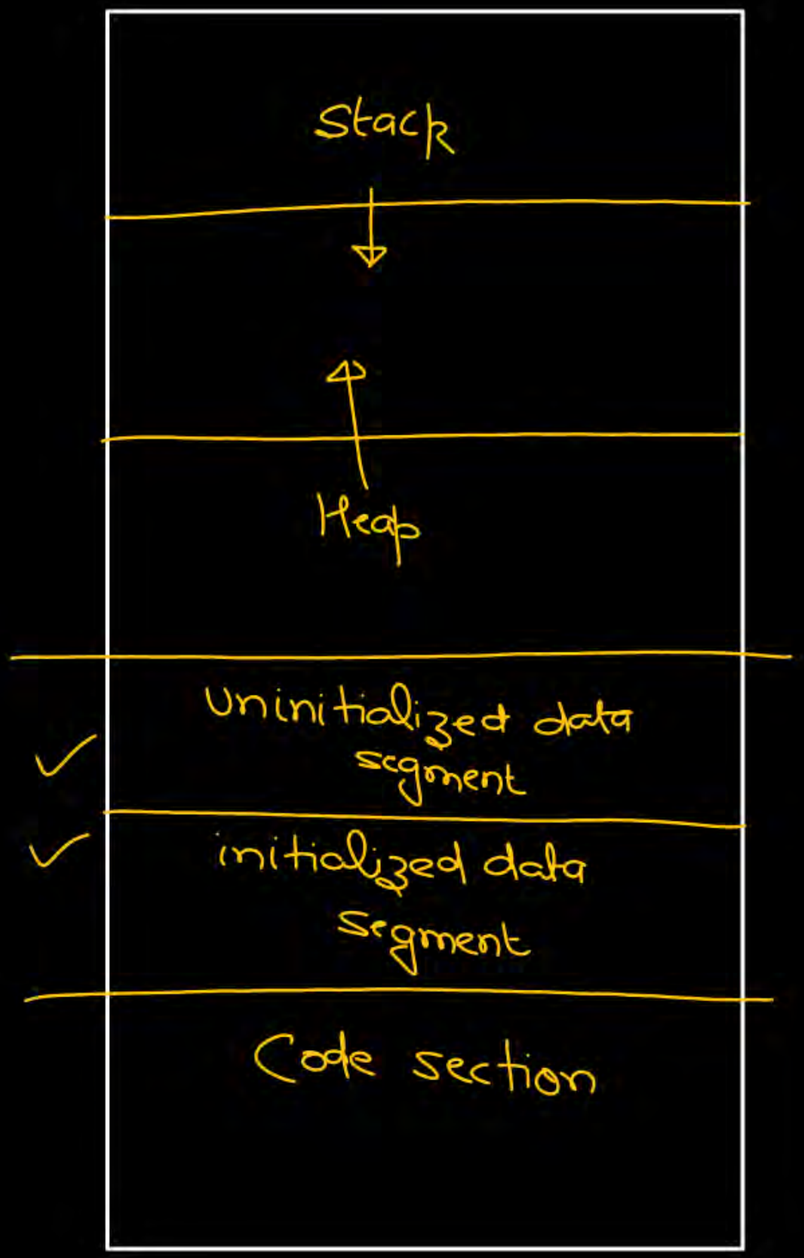
static

(i) scope : block in which they are created.

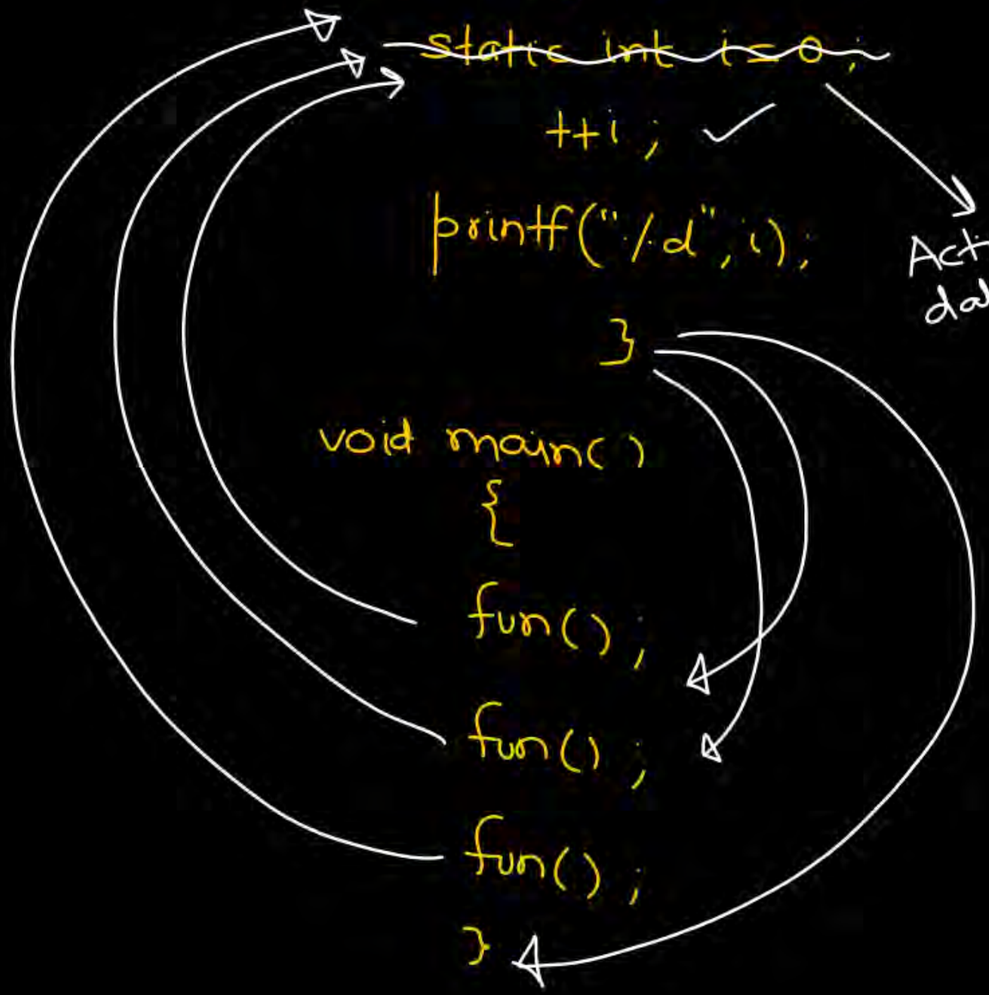
(ii) Lifetime : Program

(iii) Default value : 0

(iv) storage area : static area (data segment)



```
#include<stdio.h>
void fun( ){
    static int i=0;
    ++i ;
    printf("%d",i);
}
void main()
{
fun();
fun();
fun();
}
```



i fun | 0123

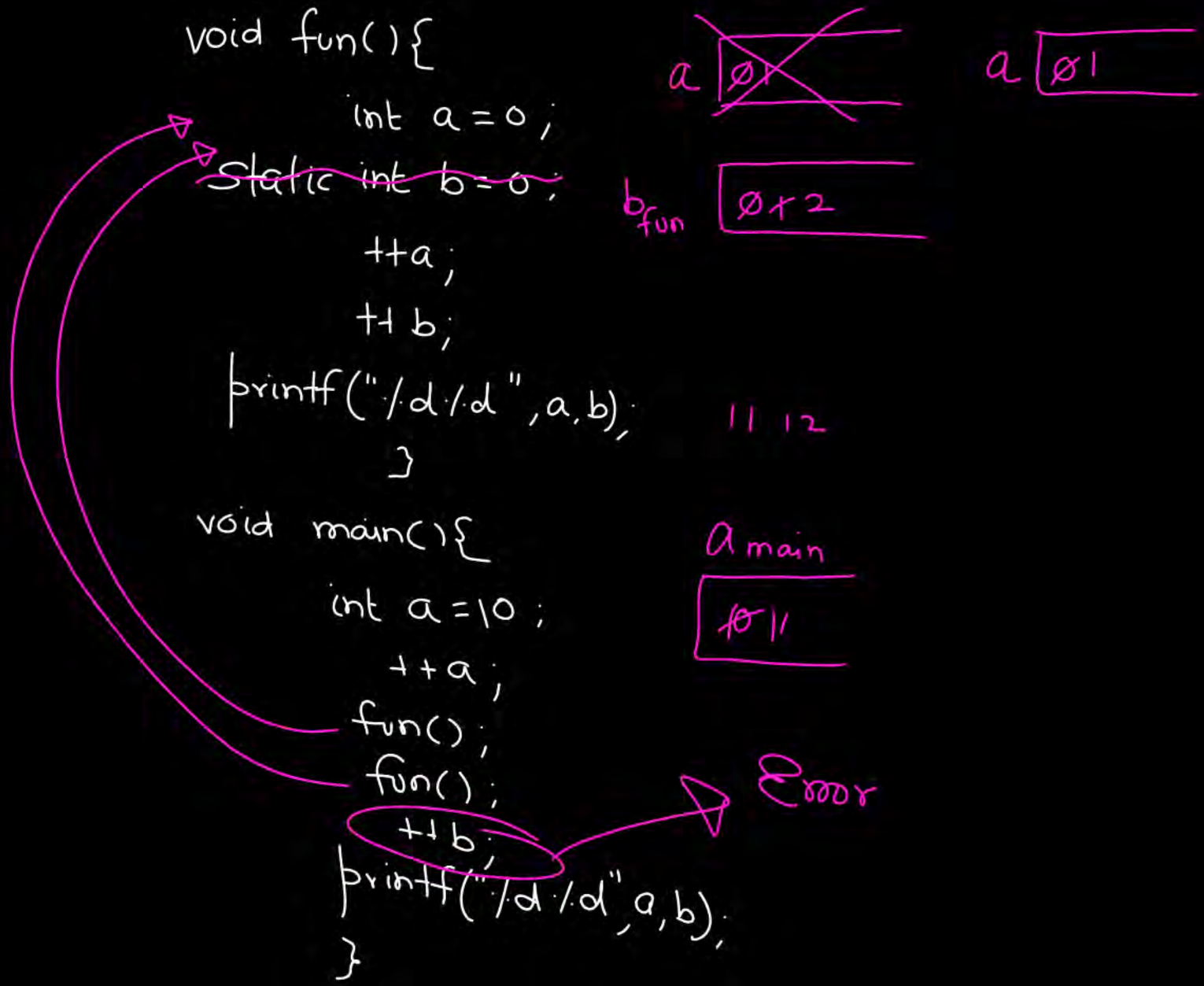
Activation record data segment ✗ ✓

1 2 3

(i) No redeclarations

(ii) Value persist b/w different function call.

(iii) They are created only once in the program.


void fun(){
int a=0;
static int b=0;
++a;
++b;
printf("%d%d",a,b);
}
void main(){
int a=10;
++a;
fun();
fun();
++b;
printf("%d%d",a,b);
}
a Ø
a Ø1
b fun Ø+2
11 12
a main
10 11
Error

purpose

Global variable

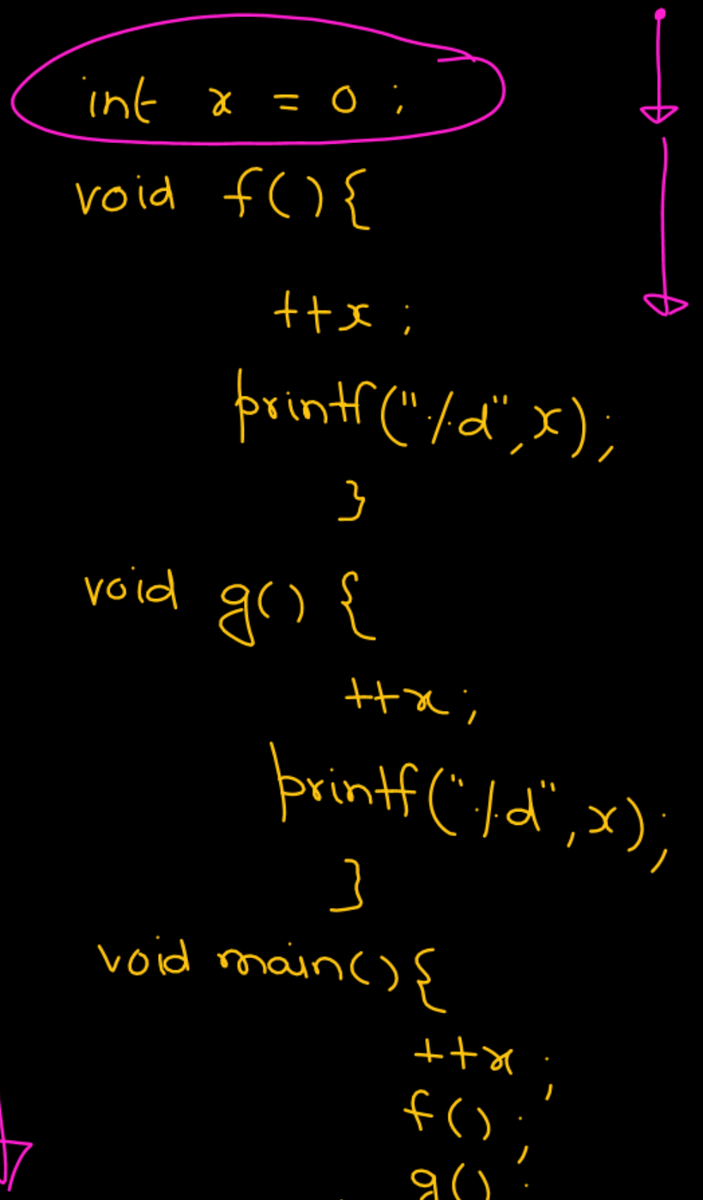
→ improvement

```
int i=10;
void f(){
}

void g(){
}
```

by default, a variable defined outside all the functions is ⇒ global variable.

```
void main(){
}
```


int x = 0;
void f(){
++x;
printf("%d",x);
}
void g(){
++x;
printf("%d",x);
}
void main(){
++x;
f();
g();
printf("%d",x);
}

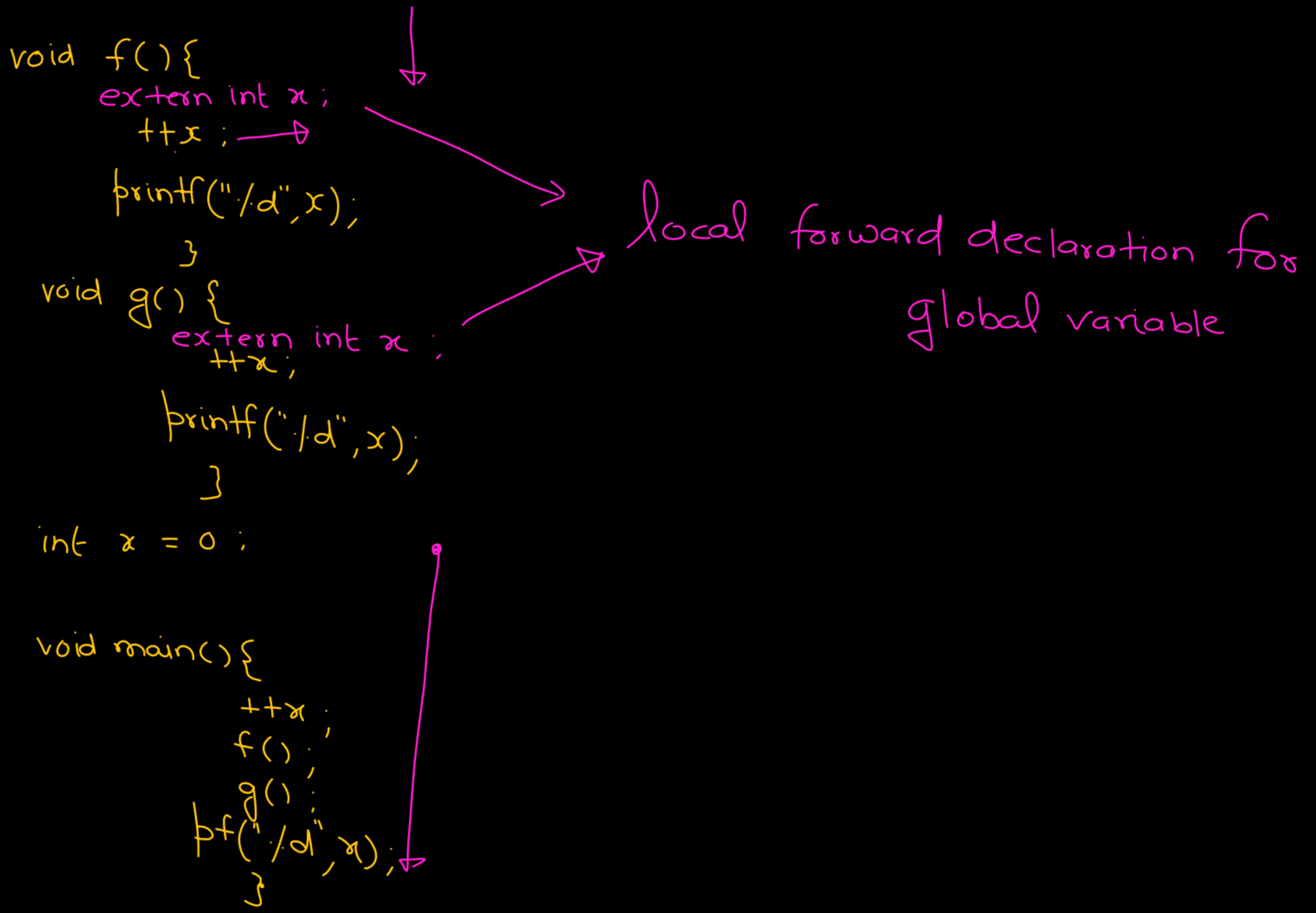
```
void f(){
    ++x;
  printf("%d",x);
    }
```

? (use) → Compiler Error

```
int x = 0;

void g() {
    ++x;
  printf("%d",x);
    }

void main(){
    ++x;
    f();
    g();
  bf("%d",x);
    }
```

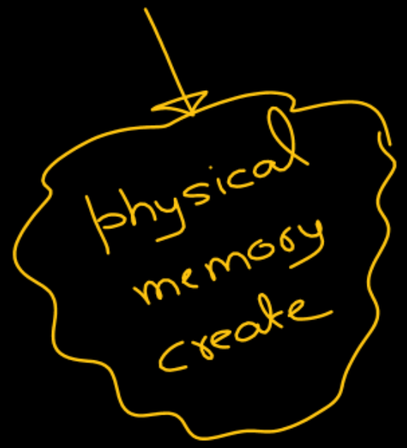

void f(){
extern int x;
++x;
printf("%d",x);
}
void g(){
extern int x;
++x;
printf("%d",x);
}
int x = 0;
void main(){
++x;
f();
g();
printf("%d",x);
}
local forward declaration for global variable

```
void f(){
    extern int x;
  ++x;
  printf("%d",x);
}
void g(){
    extern int x;
    ++x;
  printf("%d",x);
}
void main(){
    extern int x;
    ++x;
    f();
    g();
  pf("%d",x);
}
int x=0;
```

```
extern int x;
void f(){
  ++x;
  pf("%d",x);
}
void g(){
  ++x;
  pf("%d",x);
}
Void main(){
  ++x;
  f();
  g();
  pf("%d",x);
}
int x=0;
```

global forward declaration

declaration/define ⟹ diff. terms for global



```
void f(){
    extern int x;
  ++x;
  printf("%d",x);
}
void g(){
    extern int x;
    ++x;
  printf("%d",x);
}
void main(){
    extern int x;
    ++x;
    f();
    g();
  pf("%d",x);
}
int x=0;
```

```
extern int x;
void f(){
  ++x;
  pf("%d",x);
}
void g(){
  ++x;
  pf("%d",x);
}
Void main(){
  ++x;
  f();
  g();
  pf("%d",x);
}
int x=0;
```

decl

global forward declaration

define

1)

```
int j=10;
static int i = j;
```

Invalid

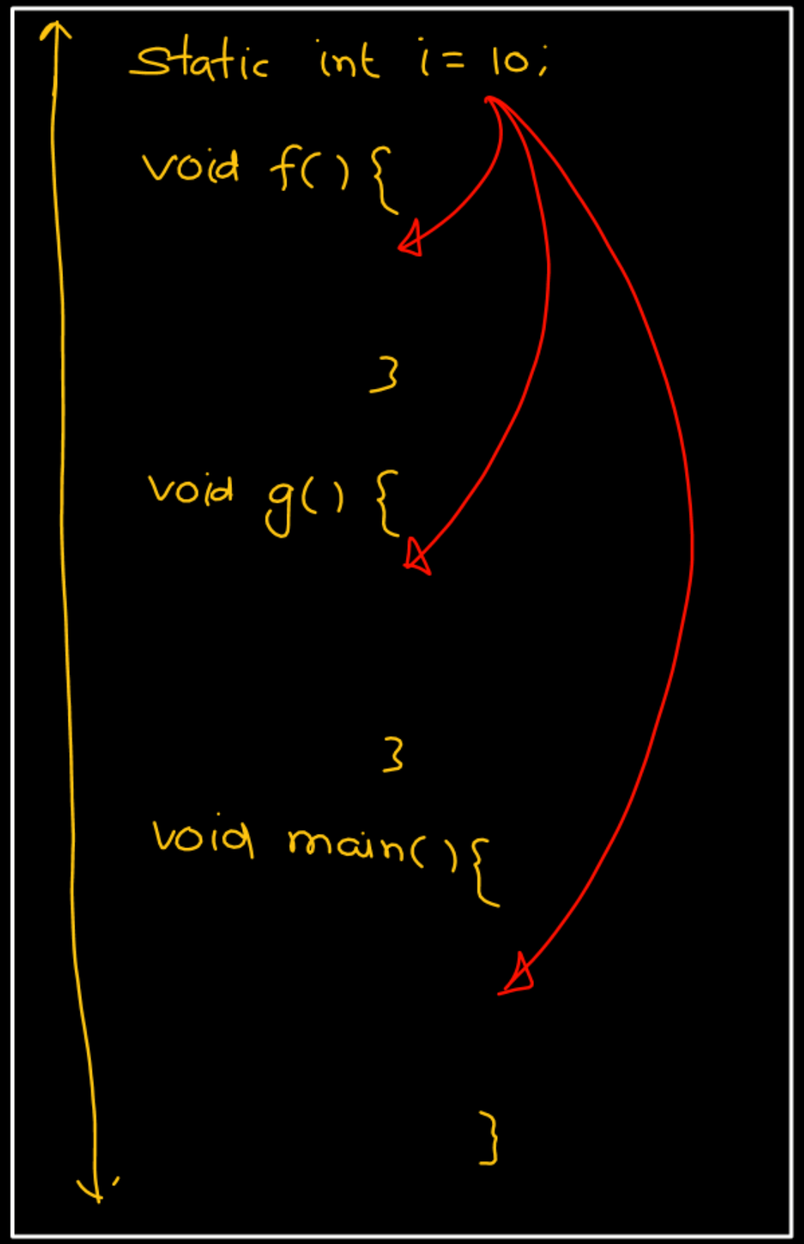
Static variable ⇒ constant/literal

initialize

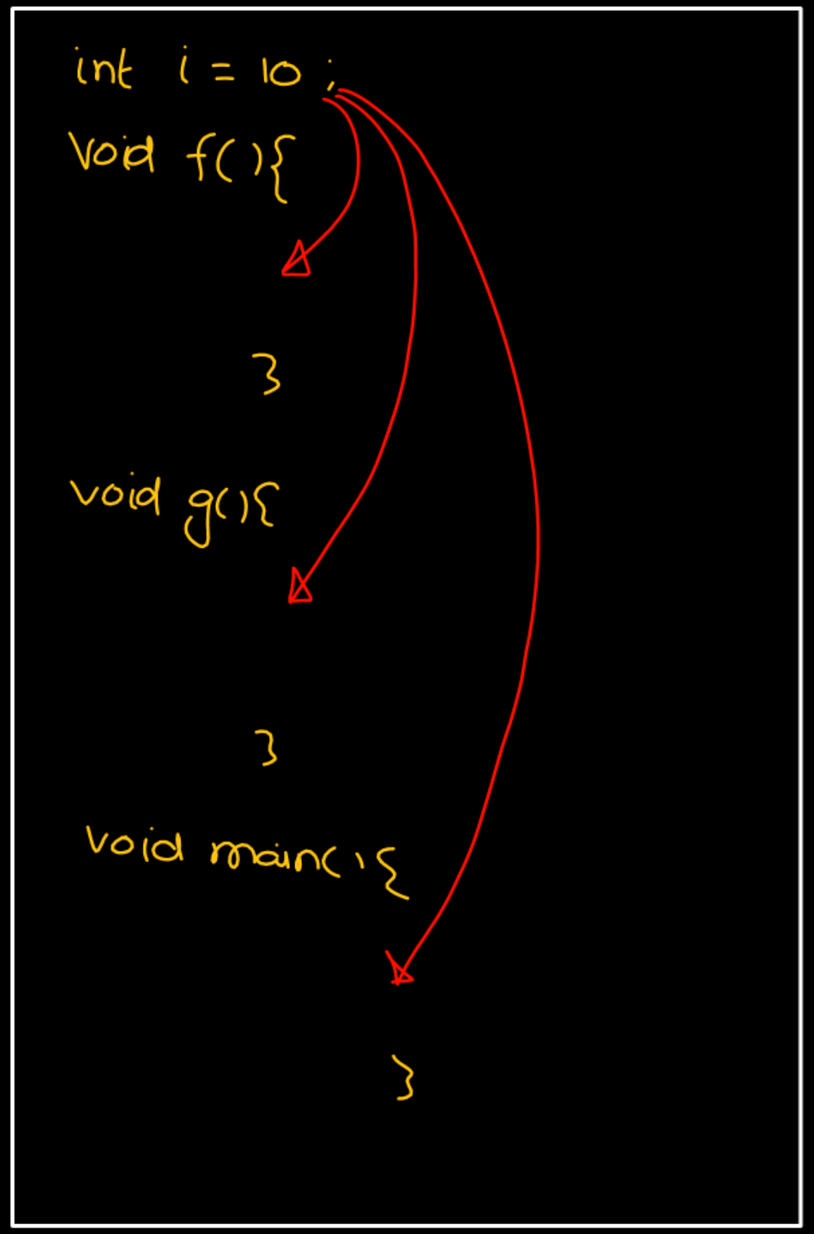
2)

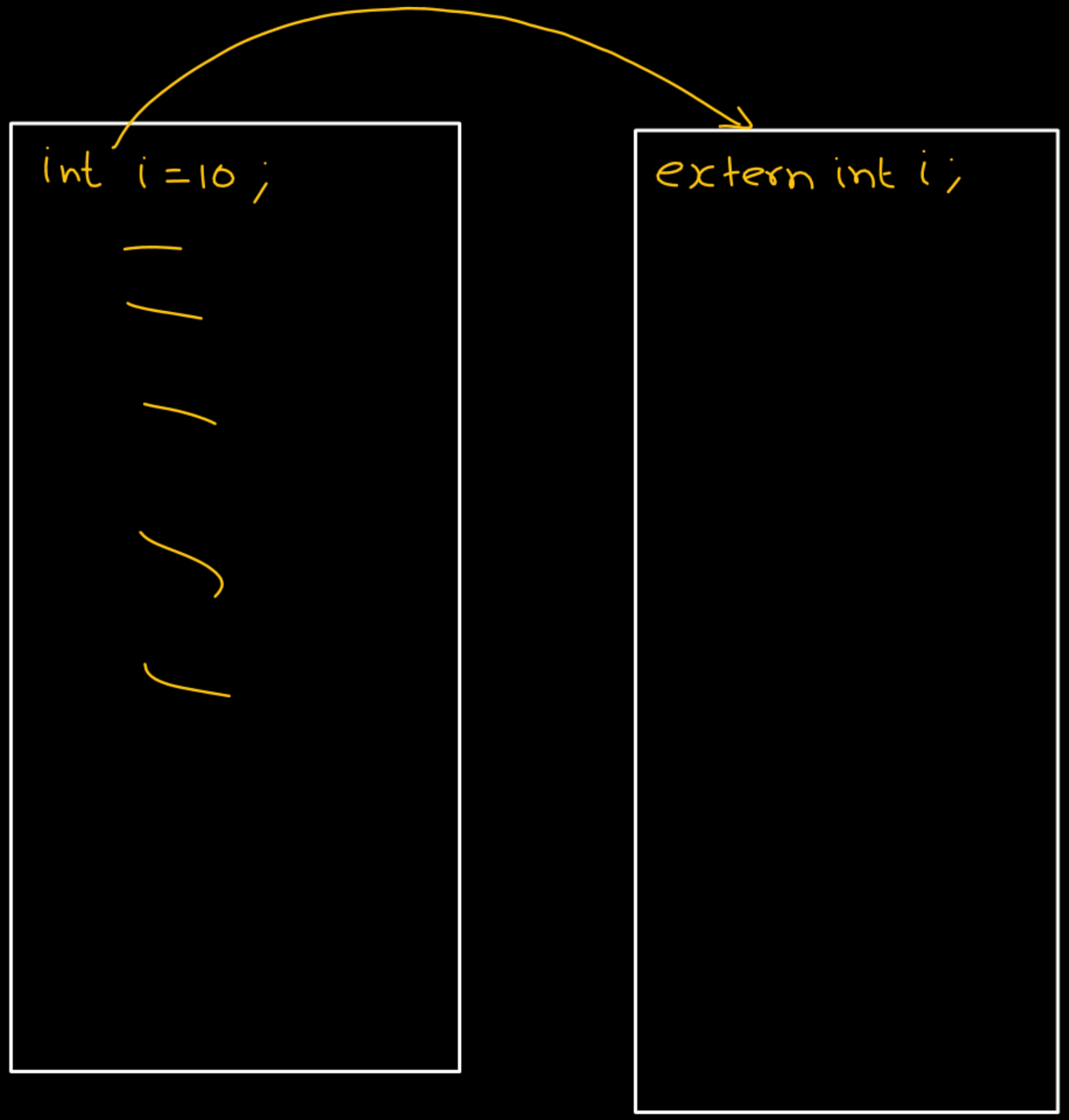
```
fun() {
        int i ;
        int i ;
      }
```

internal linkage



P.C



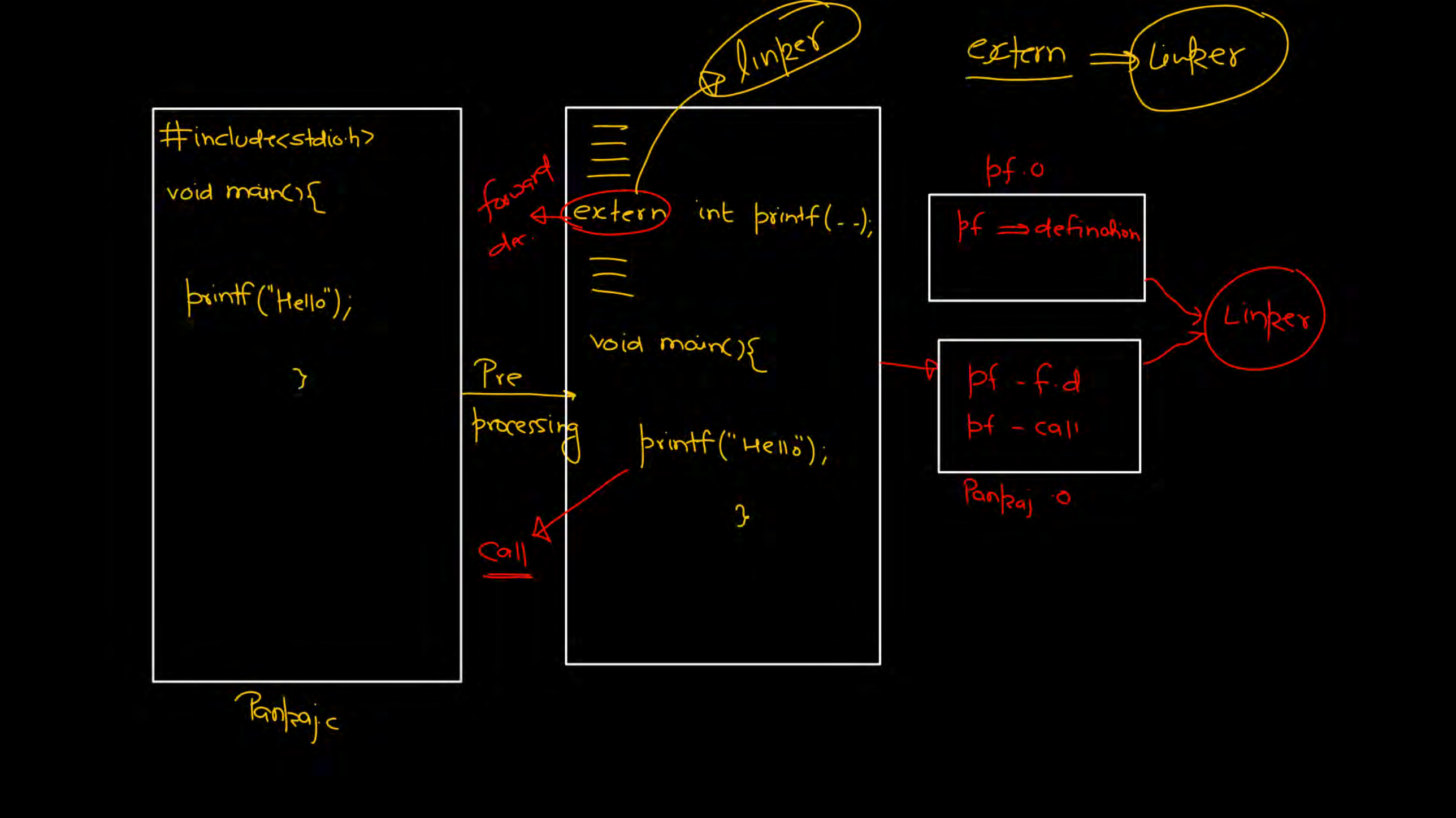


external linkage

block

file

Multifile

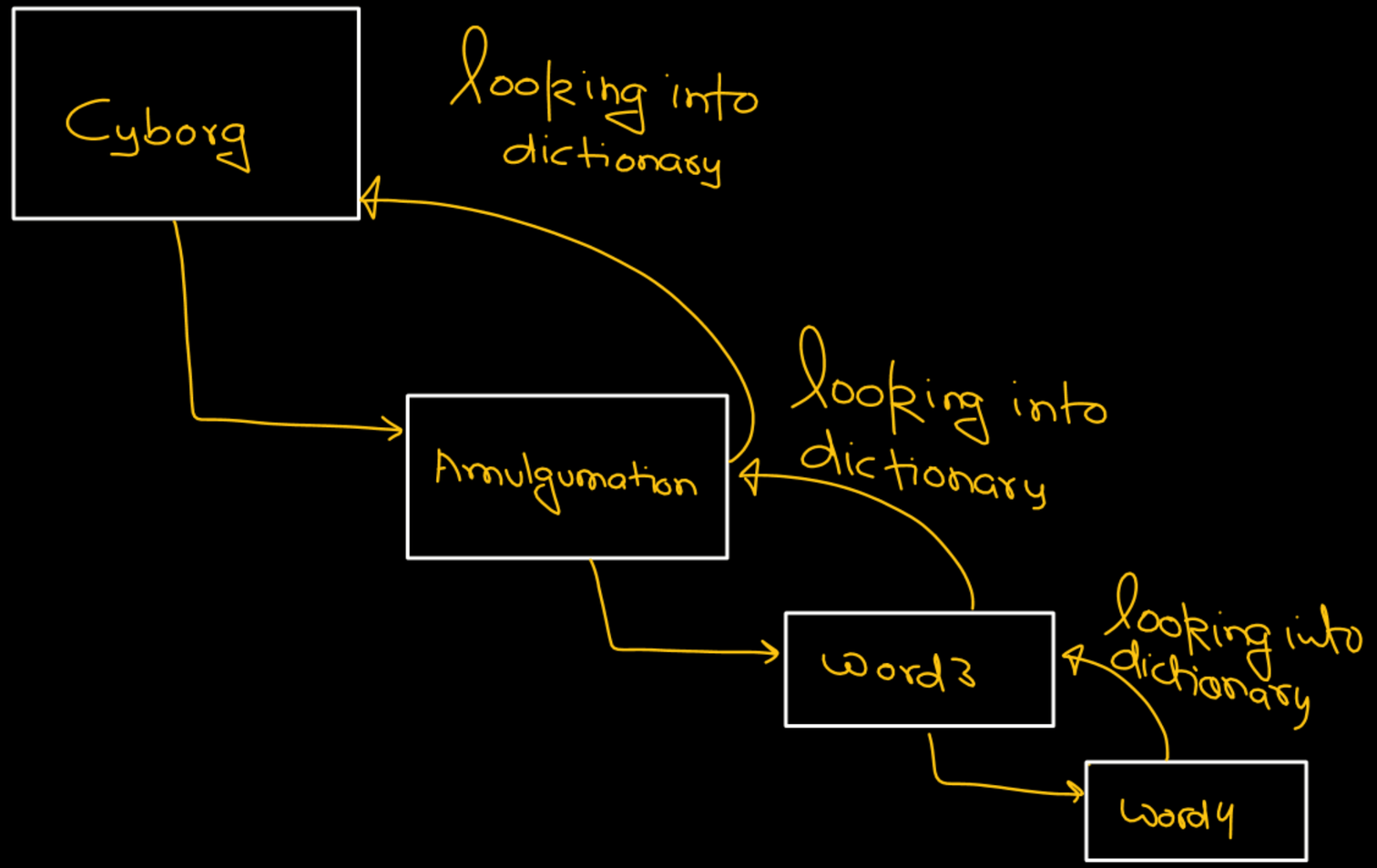

linker
extern ⇒ Linker
#include<stdio.h>
void main(){
printf("Hello");
}
Pankaj.c
forward dec.
extern int printf(--);
Pre
processing
void main(){
printf("Hello");
}
Call
pf.o
pf ⇒ defination
pf - f.d
pf - call
Pankaj .o
Linker

```
int i=10;
void f(){
  =
  =
}
void g(){
  =
  =
}
```

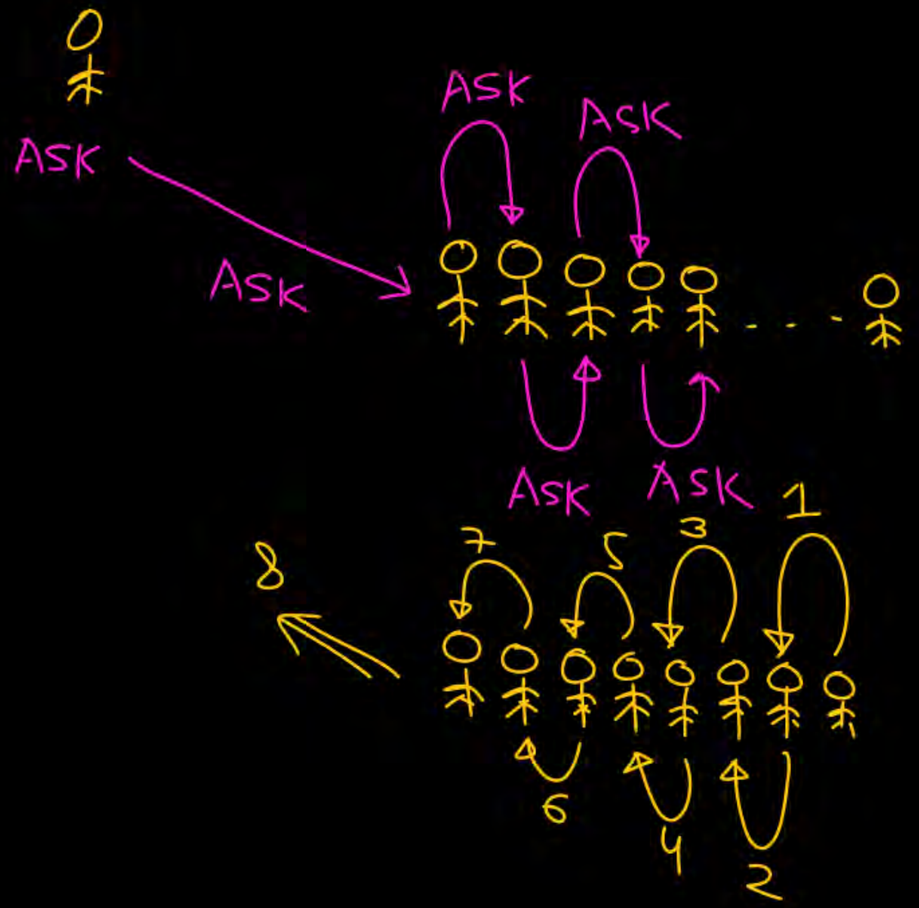
```
void main(){
}
```

Recursion

Ex1



Ex 2



What if only 1 student ?

→ No Recursion

→ Easy

→ directly answer

Atleast 2 cases

```
if( n is small )        // input is small
  {
```

* Can be answered directly.
* No recursion is needed.
* Easy problem

```
  }
else {
```

* Can not be answered directly
* Not Easy
* Recursion is needed

i/p is large

ईश्वर Call कुछ काम खुद करती & बाकी का recursion करता है।

```
  }
```

global

extern

external linkage


THANK YOU GW
SOLDIERS !