CS & IT ENGINEERING

Programming in C

Miscellaneous Topics
In One Shot

By- Pankaj Sharma sir

TOPICS TO BE COVERED

- Miscellaneous

Comma operator

1) It works as a seperator.

```
int x = 5, y = 6, z = 10;
```

OR

```
int x = 5;
int y = 6;
int z = 10;
```

② It works as an operator

```
int x;
x = (3,4,5);
```

Variable = (Exp1, Exp2, Exp3, ... Expk);

All these expression are evaluated from left to right and the final value is the right most exp. value.
Each exp is eval. and simply rejected

② It works as an operator

```
int x;
x = (3,4,5);
```

$\boxed{x \Rightarrow 5}$

Variable = (Exp1, Exp2, Exp3, . . . Expk);

All these expression are evaluated from left to right and the final value is the right most exp. value.

Each exp is eval. and simply rejected

```
int i;
i = (printf("Pankraj"), 10 + 5);
printf("%d", i);
```

i = (~~printf("Pankraj")~~ , 10 + 5);

① 6 × 15

15

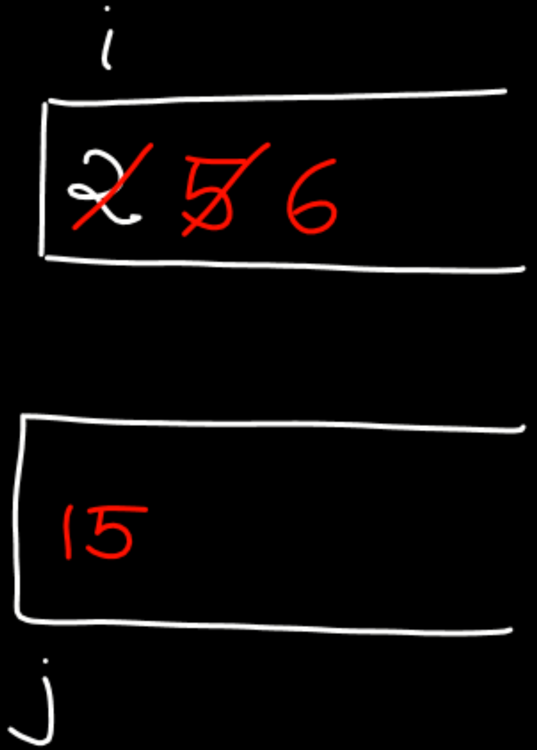
```
Pankraj15
```

not changing i

15

```
int i=2, j;
j = ( i=i+3, ++i , i+9) ;
printf("%d%d", i, j);
```

615



$6+9=\boxed{15}$

3. Least Priority

$a = 3, 4, 5;$ ✓

1) $=$ High

2) $,$ Low

$\Downarrow$

$(a=3), 4, 5;$

①

(i) $\cancel{(a=3)};$ $\quad a \; \boxed{3}$

$\cancel{3}, \cancel{4}, 5;$ ✓

```
int x=5, y=10, z=10;
```

OR

```
int x = 5;
int y = 10;
int z = 10;
```

a $\boxed{3}$

Valid

$(a = 3), 4, 5 ;$

$\Downarrow$

$\cancel{3}, \cancel{4}, 5 ;$ ✓

```
void main(){
    12;        Exp ✓
    17.38;     Exp ✓
}
```

$\cancel{3}, \cancel{4}, 5 ;$ // Expression

value 5

Union

* Just like structure, union is also a user defined data type.

* union is the keyword which is used to create/define user defined data type.

```
union A {
    int x;
    char y;
};
```

member

```
struct P {
        int x;      → 4 byte
        char y;     → 1 byte
    };
void main(){
    struct P s;
    printf("%u", sizeof(s));        5

    }
```

√All members of a structure variable get its individual memory space.

```
struct P {
        int x;      → 4 byte
        char y;     → 1 byte
      };

void main(){
        struct P  s;
        printf("%u", sizeof(s));

        }
```

✓All variable of a union variable $\boxed{\text{share}}$ common memory space

```
union Q {
        int x;  → 4
        char y; → 1
     };

void main(){
        Union Q  u;
  ④    pf("%u", sizeof(u));
        }
```

$\text{sizeof}(u) = \max(4, 1) = \textcircled{4}$

assume
char 1 byte
int 4 byte
float 8 byte

```
union Q {
    char x ;
    int y ;
    float z ;
};

void main() {
    union Q u ;
    pf("%u", sizeof(u));
}
```

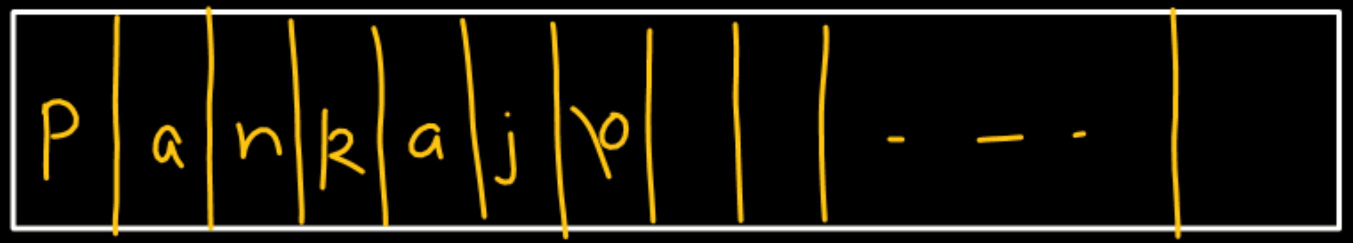
$\max(1,4,8)$

$= \boxed{8}$

```
char name[20];

pf("Enter ur name");

sf("%s", name);
```



Pankaj↵

Pankaj

```
char name[20];
pf("Enter ur name");
sf("%s", name);
```

Pankaj sharma↵

O/P: Pankaj

```
gets(name);
```

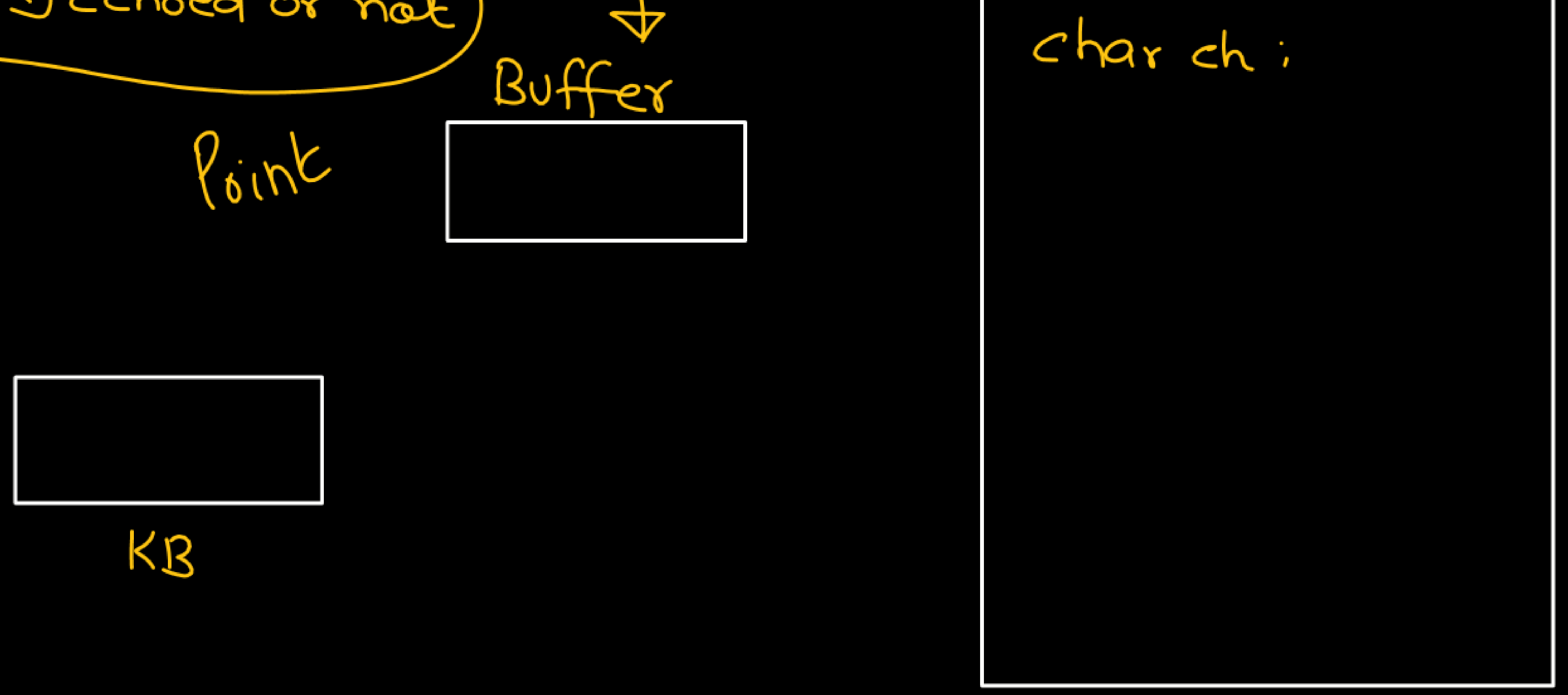
O/P: Pankaj sharma

getchar(), getch(), getche()

1) Buffered or not

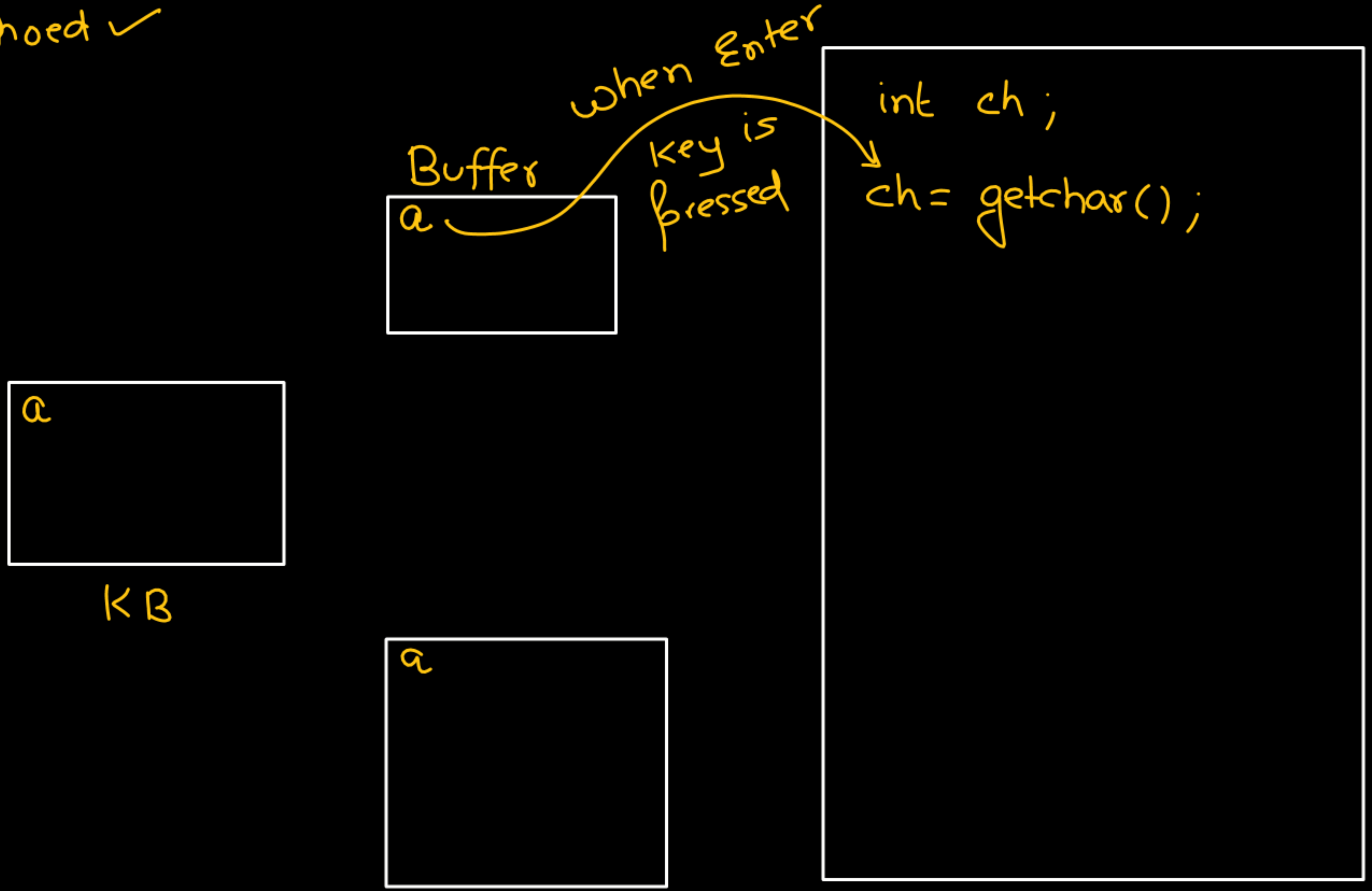
2) Echoed or not

Point



Buffered ✓
Echoed ✓

getchar()



$$\begin{tikzcd} & \underbrace{\text{scoping}} \arrow[dl] \arrow[dr] & \\ \checkmark\ \text{Static scoping} & & \text{Dynamic scoping} \end{tikzcd}$$

C, C++, Java, ...

* Run time

scope related decisions

$\Rightarrow$ compile time

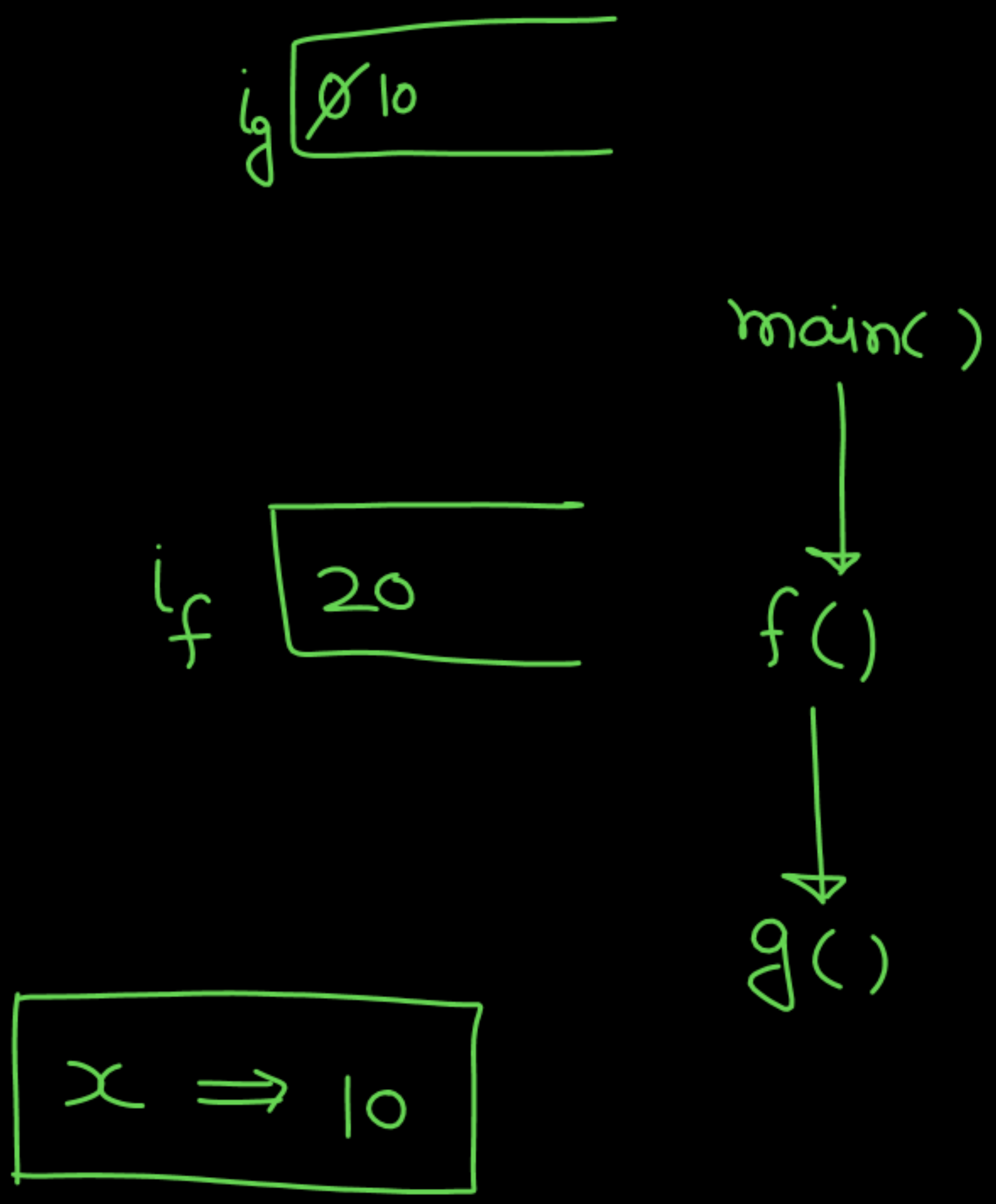
1. Consider a program in a hypothetical lang. that allow global variable and a choice of static scoping & dynamic scoping.

let

x : value under static scoping

y : value under dynamic scoping

```
int i ;
Program main() {
        i = 10;
        call f();
        }
Procedure f() {
        int i = 20;
        call g();
        }
Procedure g() {
        print(i);
        }
```



global

```
int i;
Program main() {
    i = 10;
    call f();
}
Procedure f() {
    int i = 20;
    call g();
}
Procedure g() {
    print(i);
}
```

global

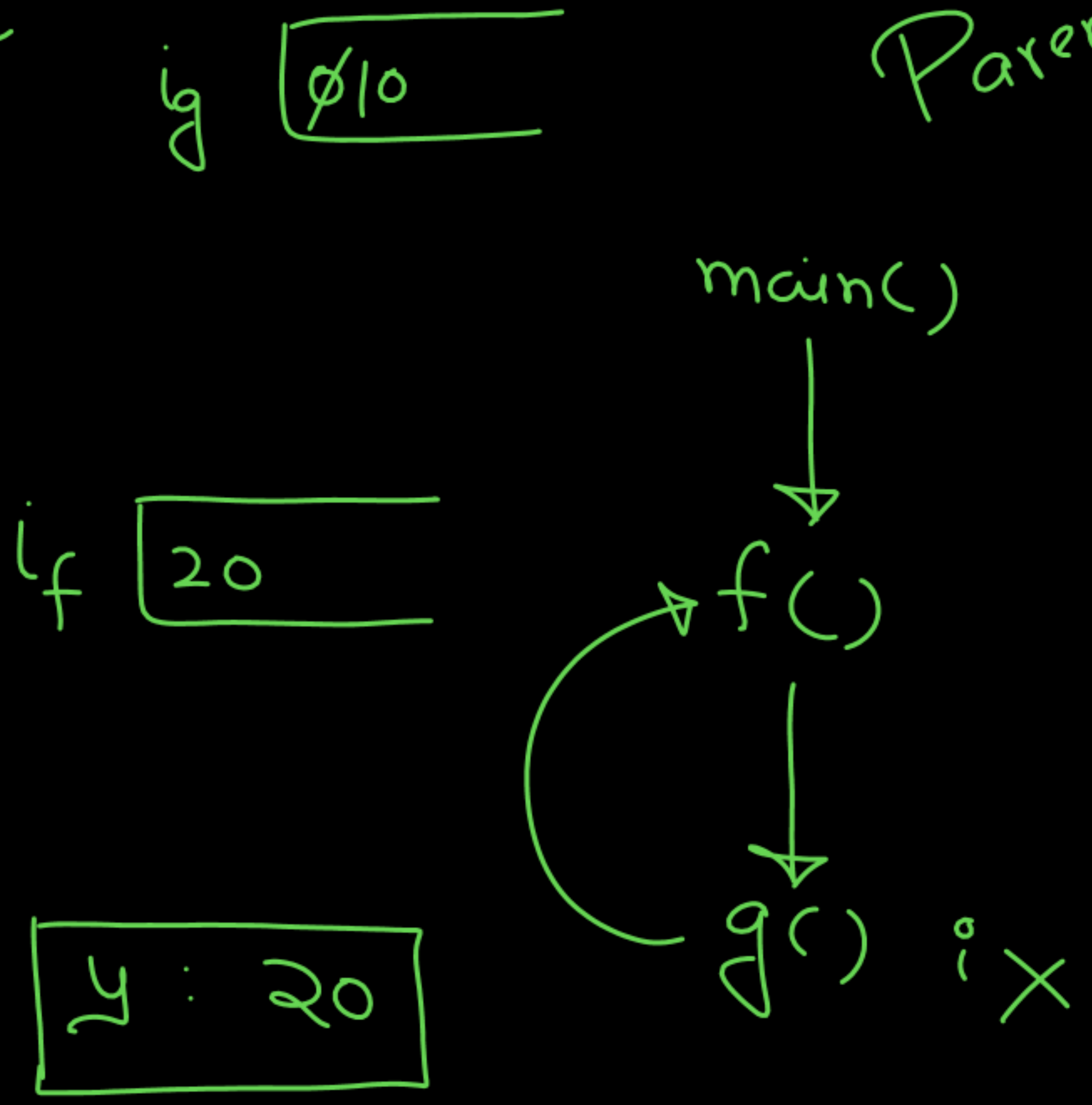
local

कौनसा i है

i_g

Dynamic scope

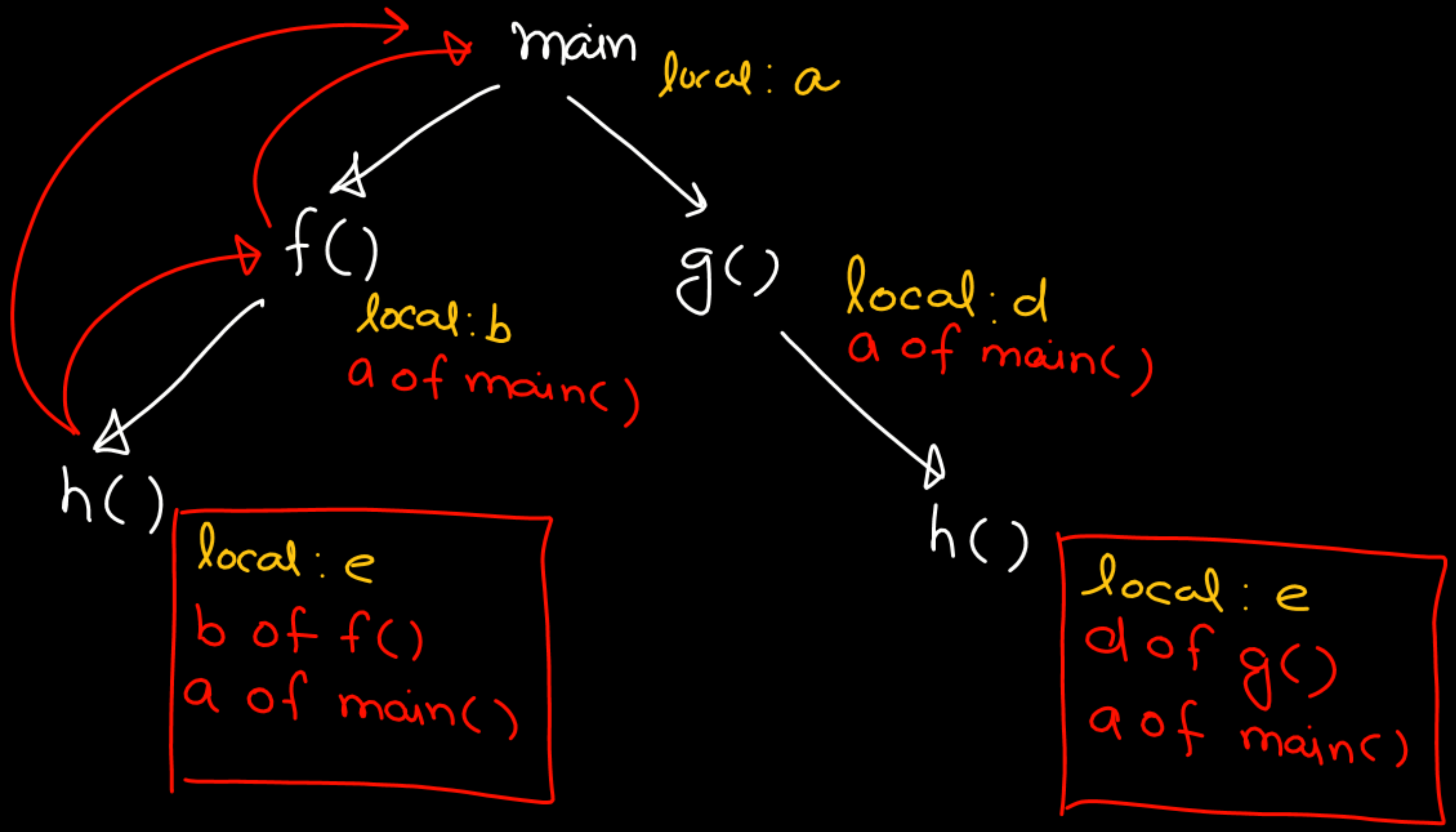
Parent function



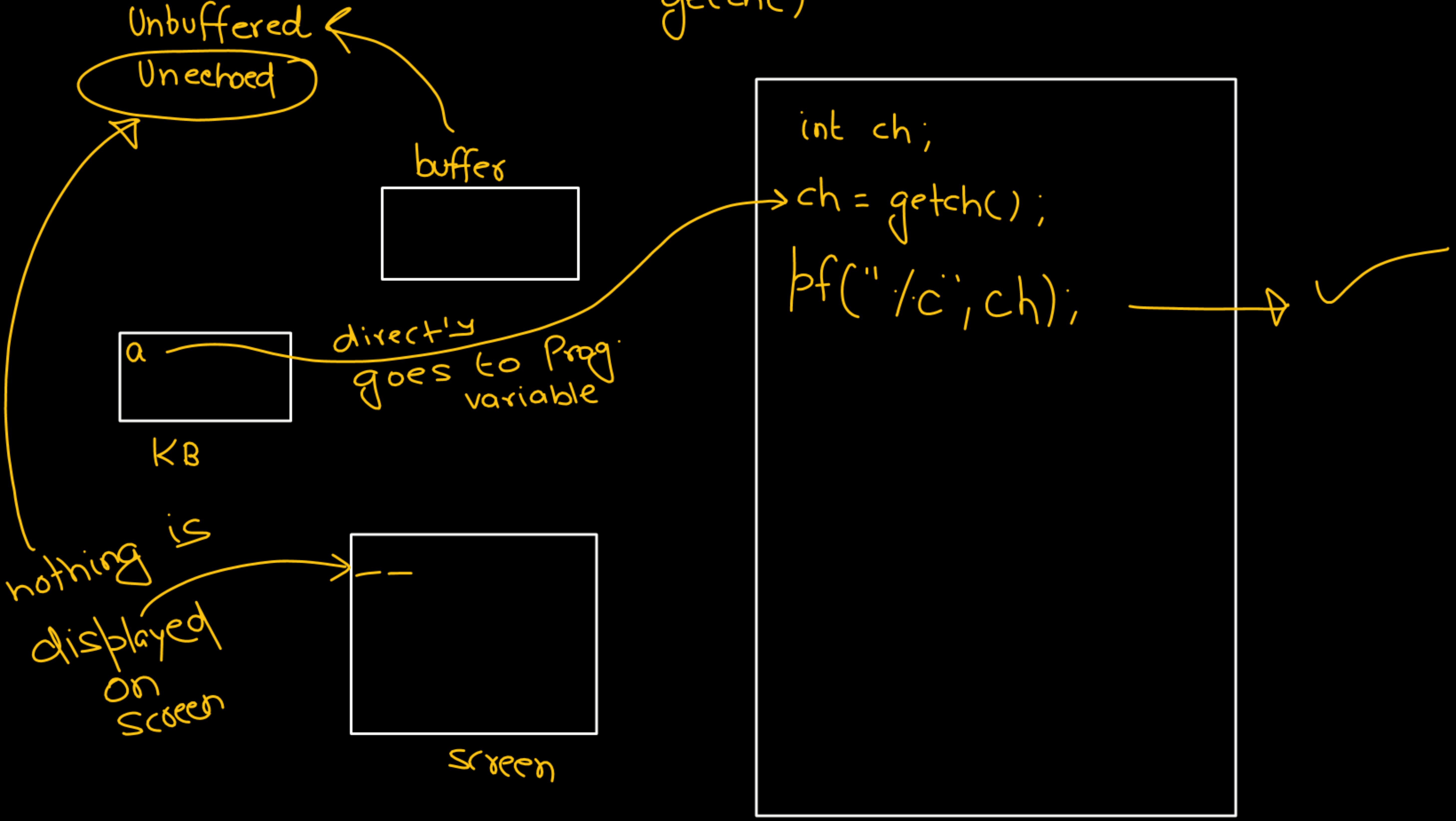
```
int i ;
Program main() {
        i = 10;
        call f();
        }
Procedure f() {
    int i = 20;
    call g();
    }
Procedure g() {
    print(i);
    }
```

Dynamic scoping

$$\begin{tikzcd}
\text{main} \arrow[dd] & \\
 & \text{local: c} \\
f(\,) \arrow[ddd] & \\
 & \text{local: b} \\
 & \text{c of main( )} \\
g(\,) & \\
 & \text{local: a} \\
 & \text{b of f( )} \\
 & \text{c of main( )}
\end{tikzcd}$$


main
local : a
f()
local : b
a of main()
g()
local : d
a of main()
h()
local : e
b of f()
a of main()
h()
local : e
d of g()
a of main()

getch()

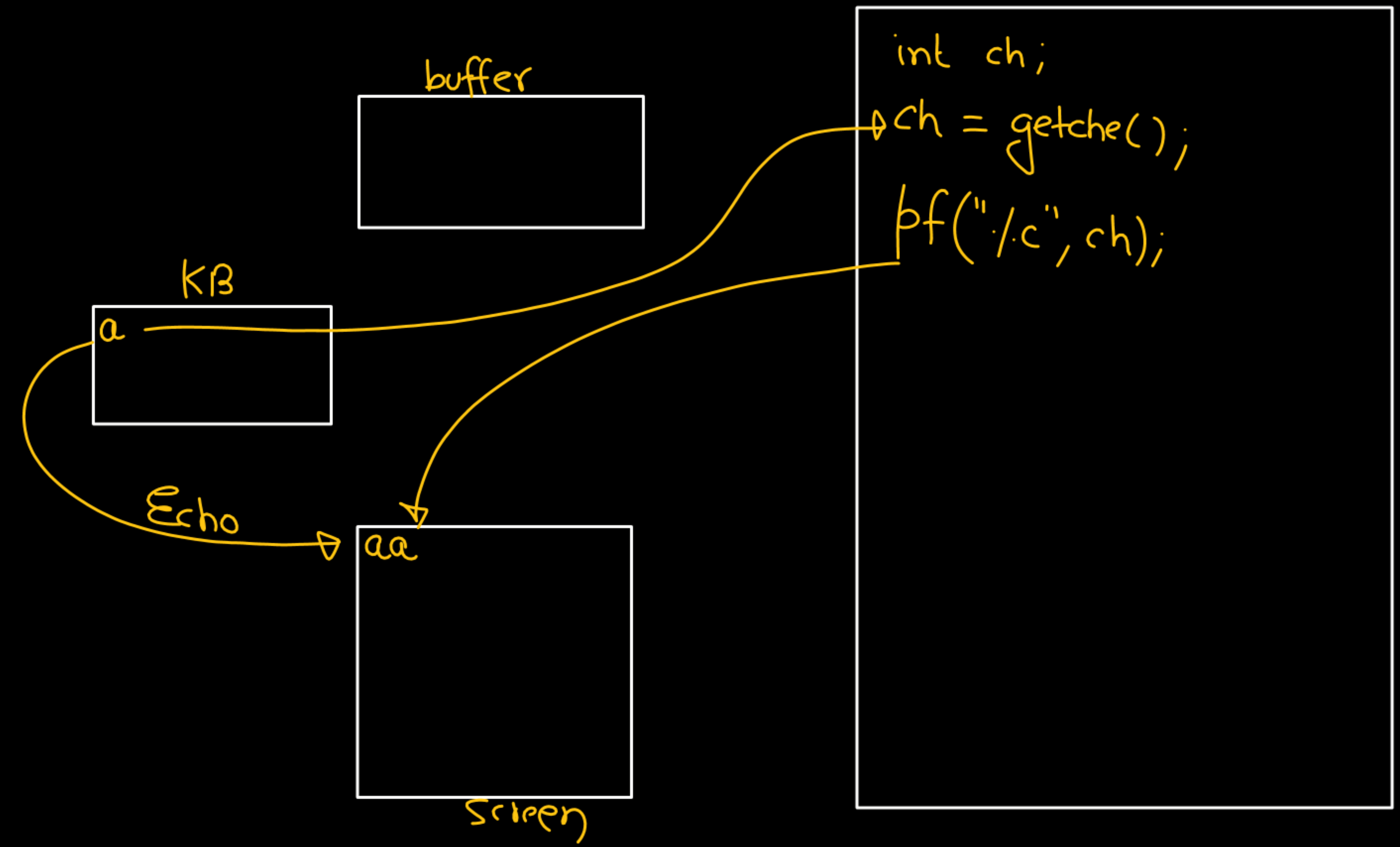


```
int ch;
ch = getch();
pf("%c", ch);
```

Unbuffered
Unechoed
buffer
a
KB
directly goes to Prog. variable
nothing is displayed on screen
--
screen

Unbuffered ✓
Echo ✓

getche



```
int ch;
ch = getche();
pf("%c", ch);
```

3Subject

1. Sizeof
2. Preprocessor
3. Questions

Saturday - DS

- arrays
- Pointer
- structure

L.L ✓
Trees ✓
Stq

THANK YOU GW SOLDIERS !