CS & IT ENGINEERING

Programming in C

Functions and Storage Classes
Lec- 02

By- Pankaj Sharma sir

TOPICS TO BE COVERED

- Storage Classes

Storage class

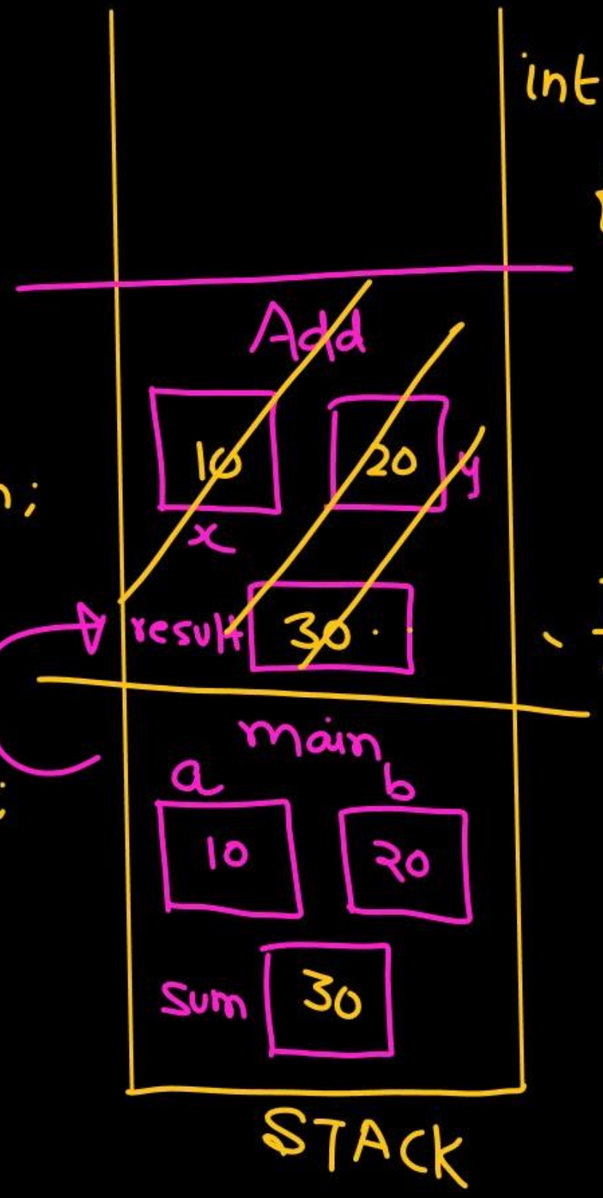
1. scope : part of code in which a variable is visible.
2. Lifetime : Duration (Active/Alive)
3. default value : If we don't initialize a variable then what is its default value.
4. storage Area : Where a variable is stored.

```
#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;
}
```

① block : {

block

}

② File

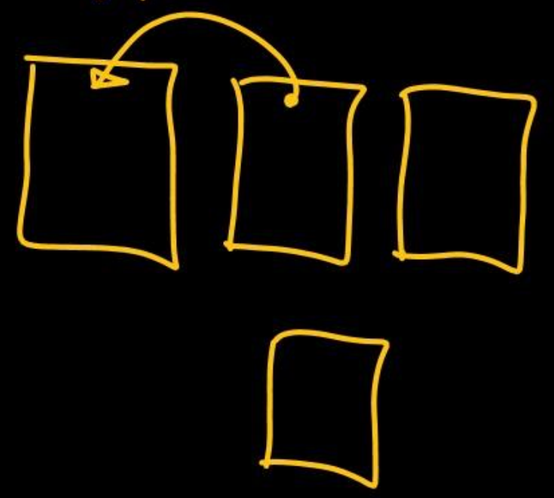
```
void f1(){
        int z;
        }
void f2(){
        int y;
        }
void main(){
        int x;
        }
```

Pankaj.c

③ Multiple files

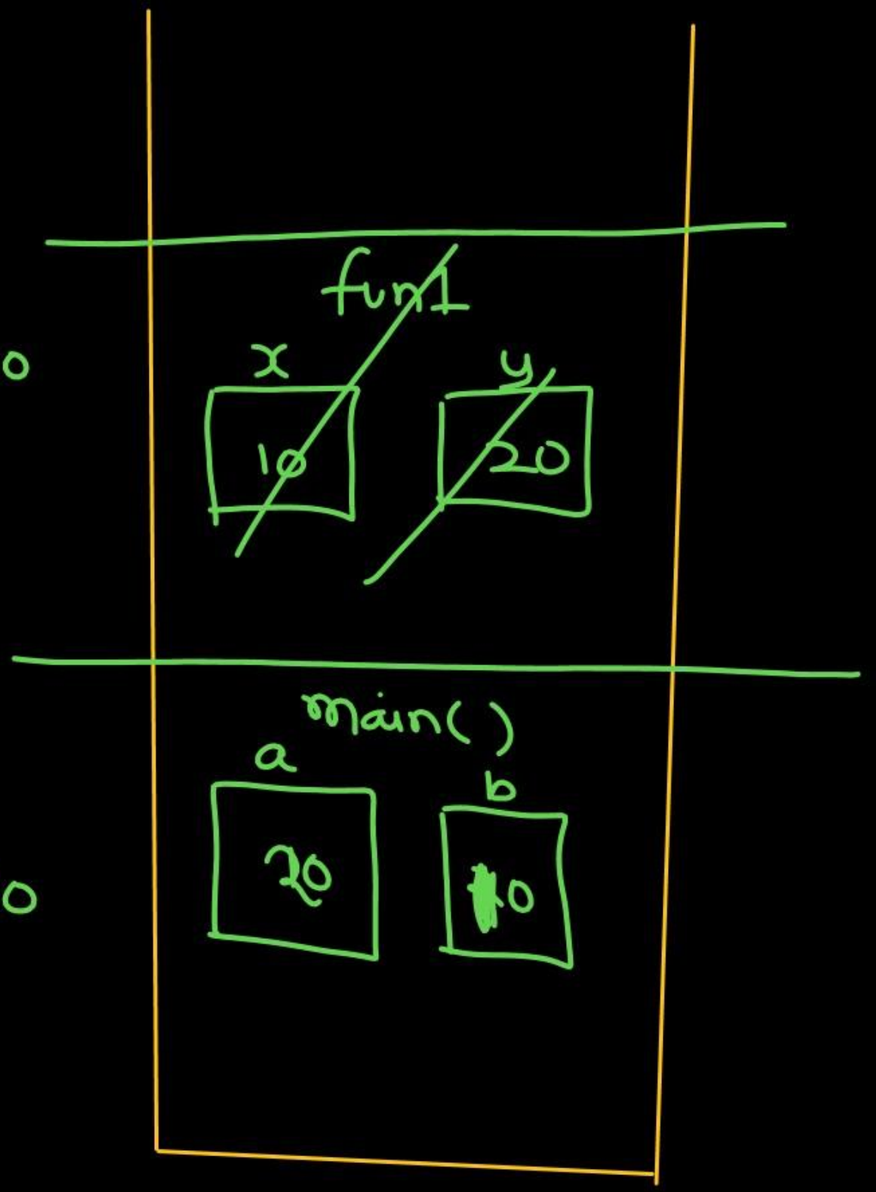
project

A



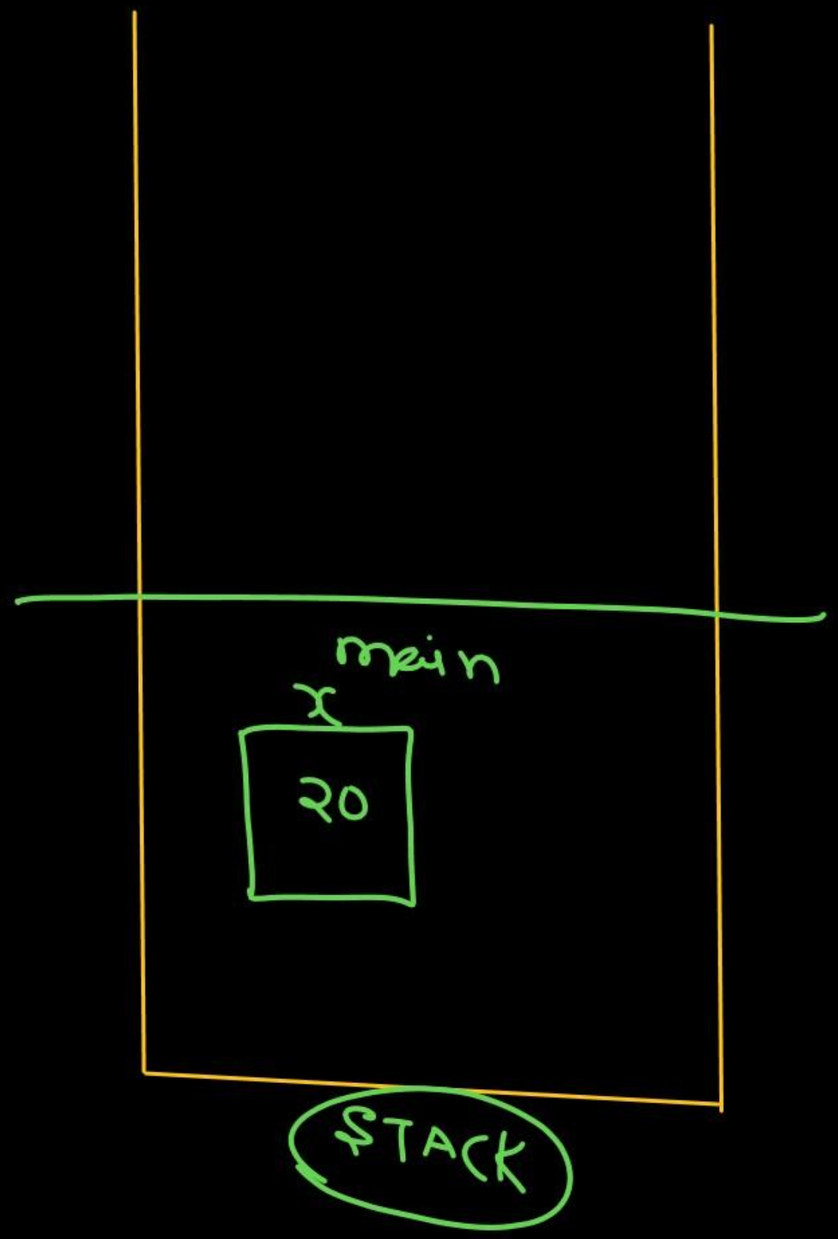
```
void fun1(){
    int x=10, y=20;
        printf("%d", x+y);  → 30
    }

void main(){
    int a=20, b=10;
    printf("%d", a+b) ⇒ 30
    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.

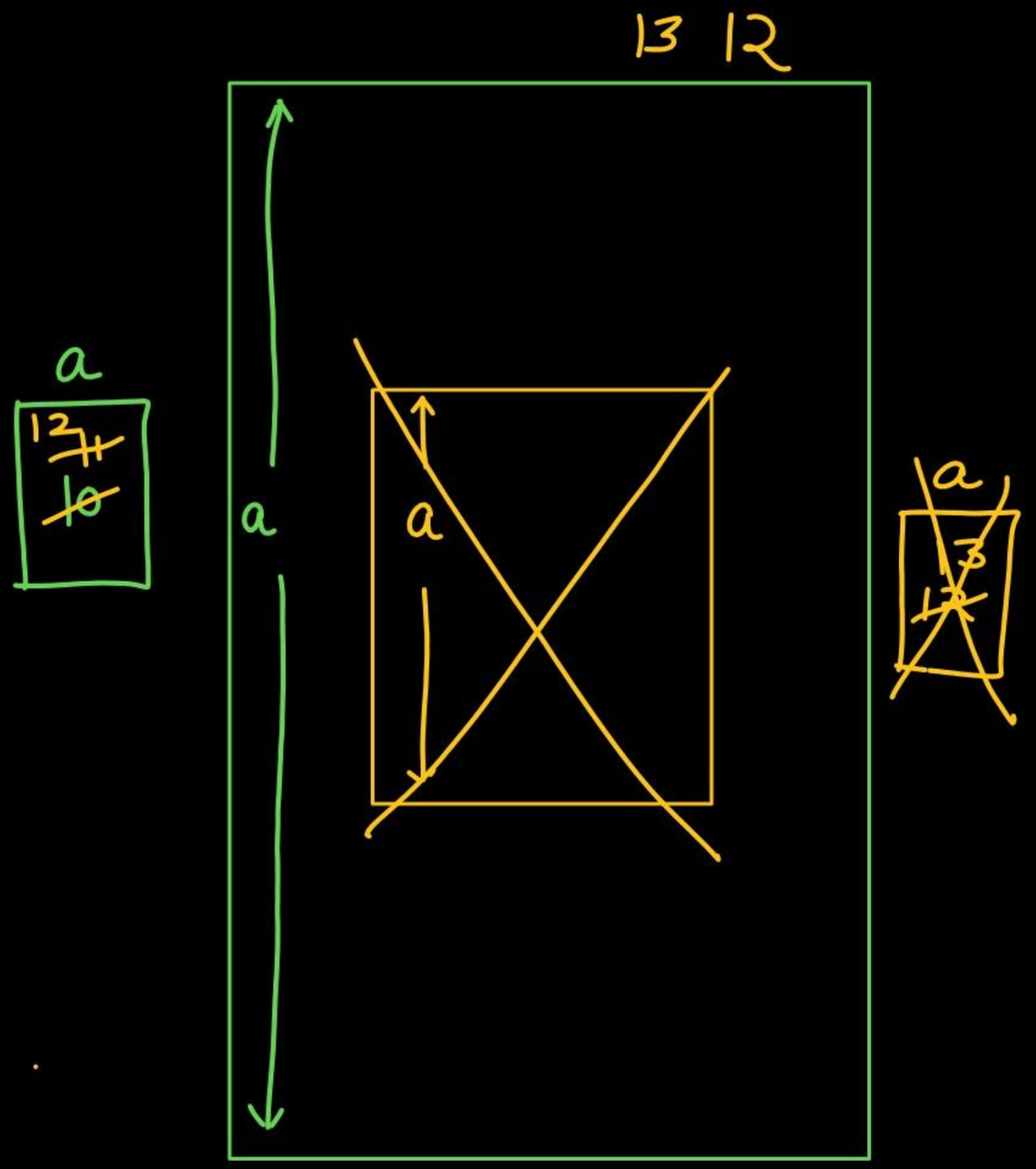
default value : Garbage

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);
}
```



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

Main scope

Sub-scope

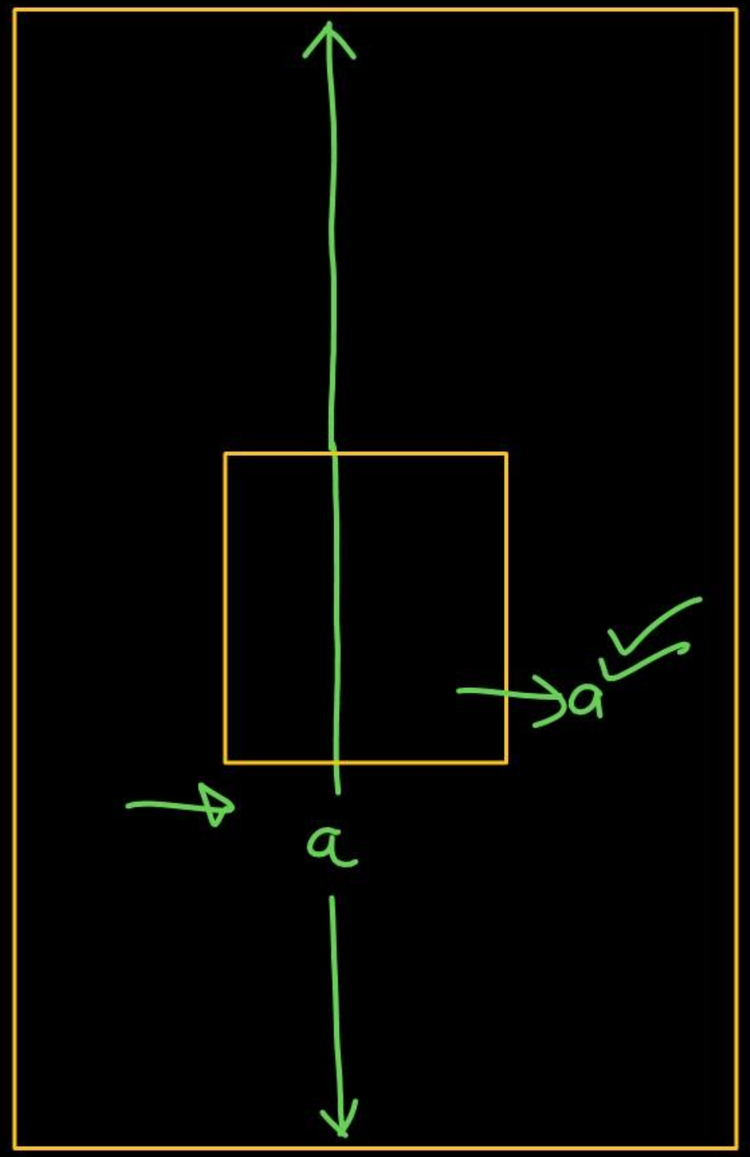
a

~~12~~ ~~11~~ ~~10~~ 13

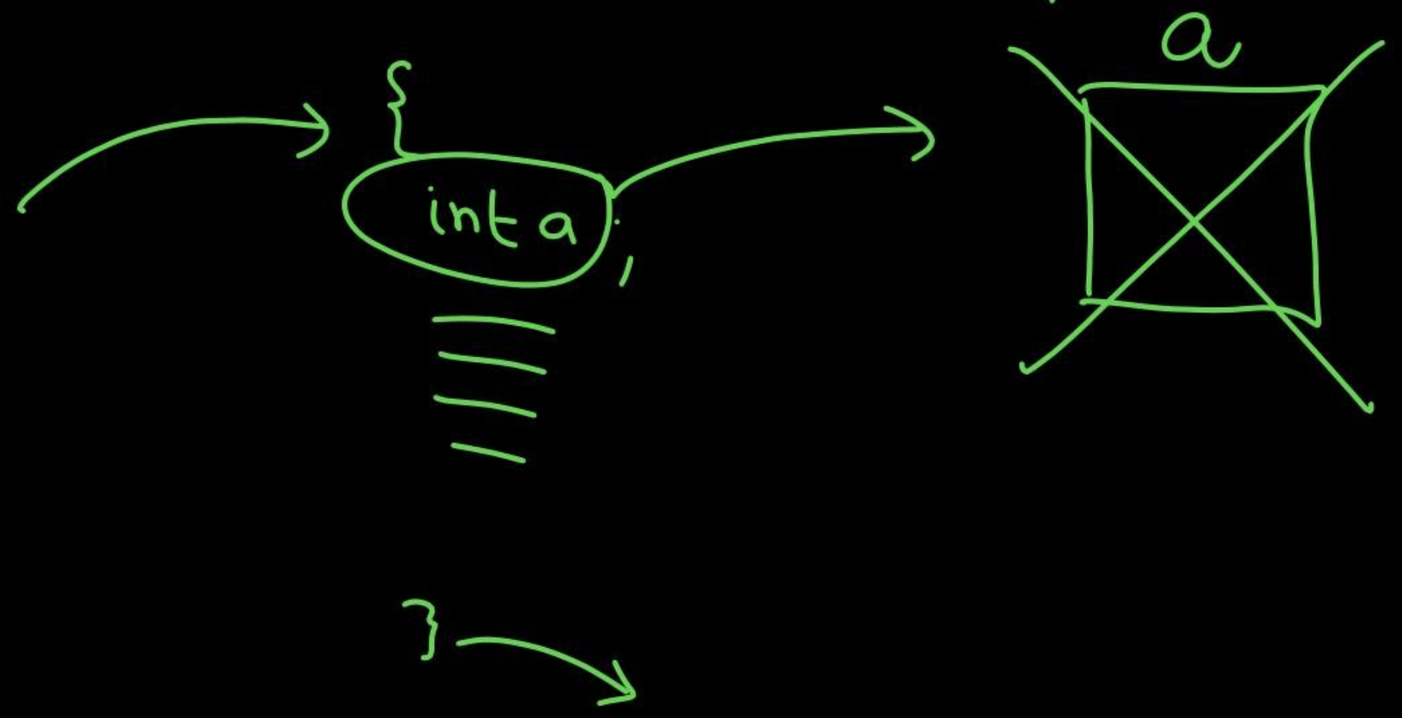
12

13

12 13



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



1. for auto ⇒ 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.

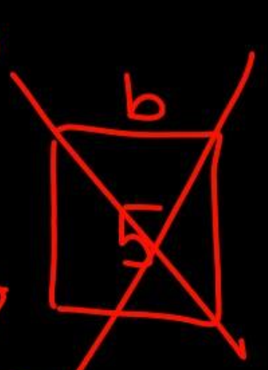
12 18

```
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);
}
```

a

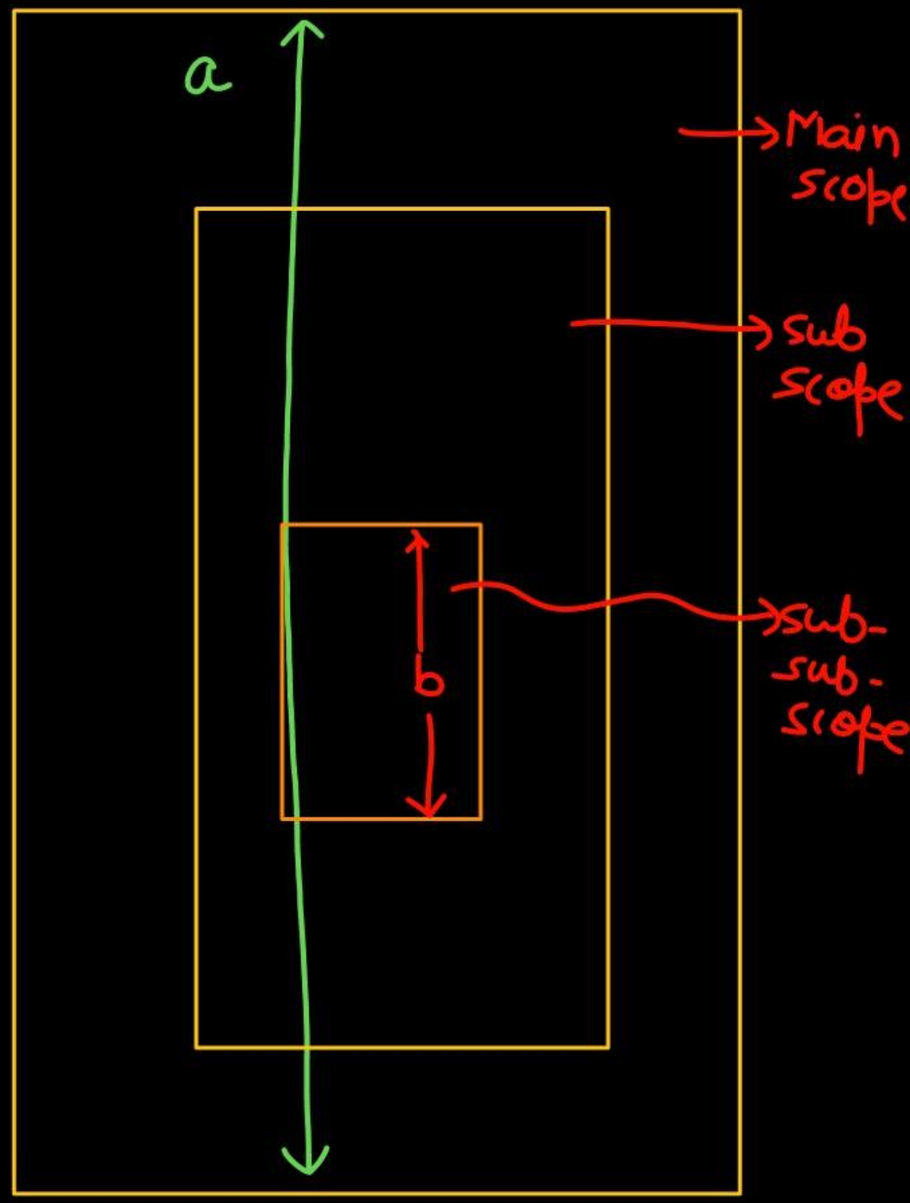
auto

b

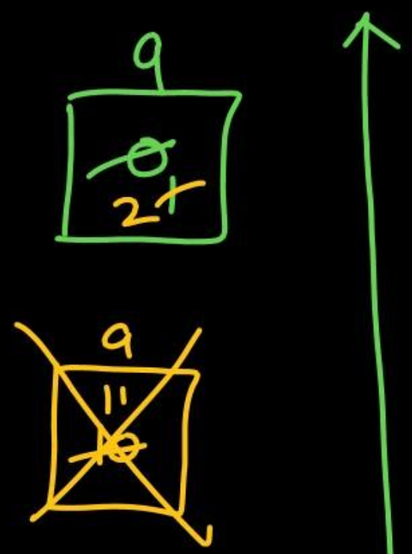


Error

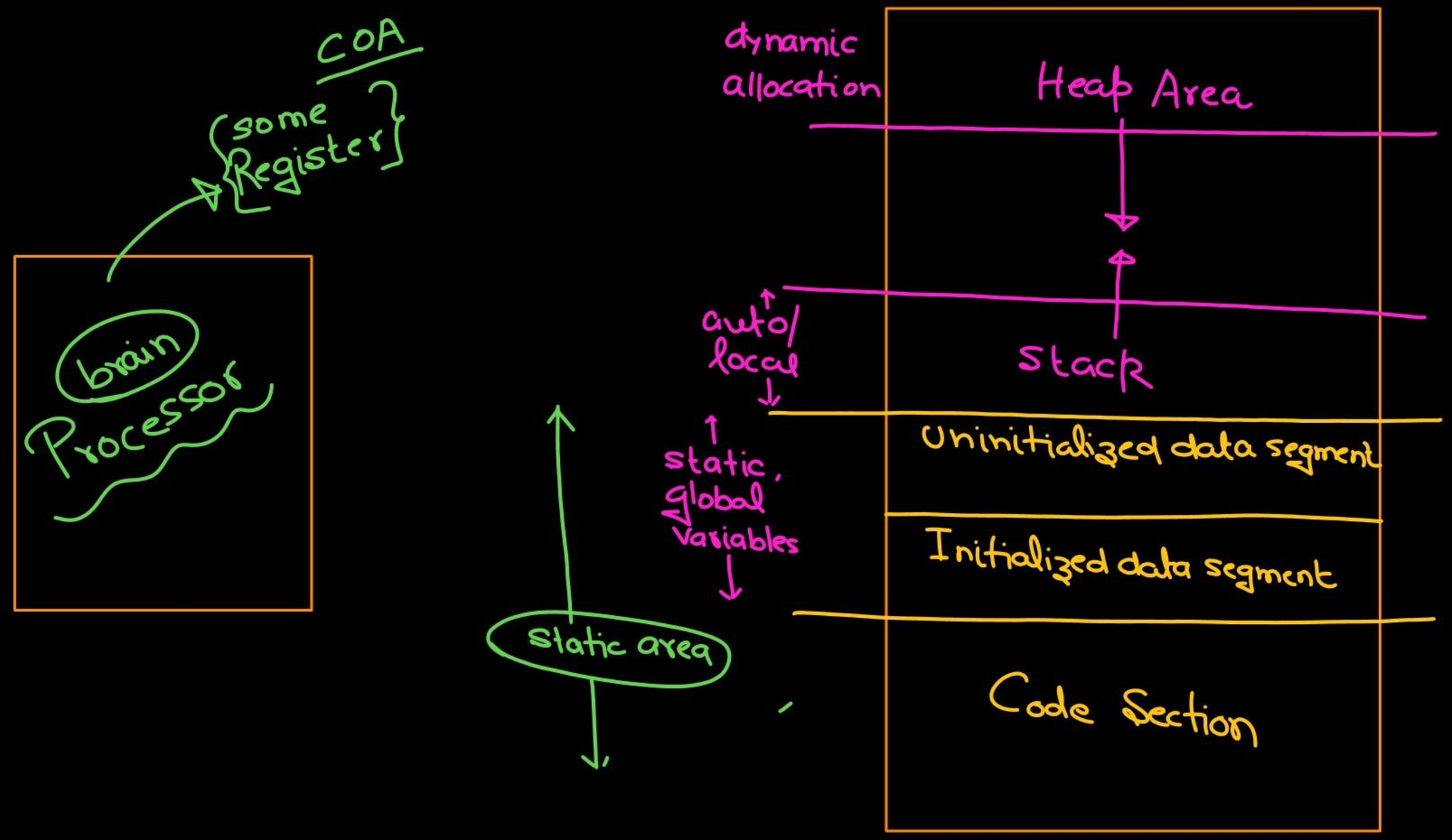
Ud ke baat Padegi



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

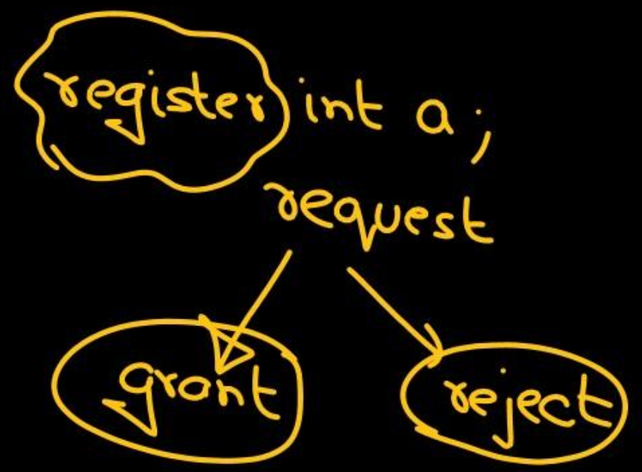


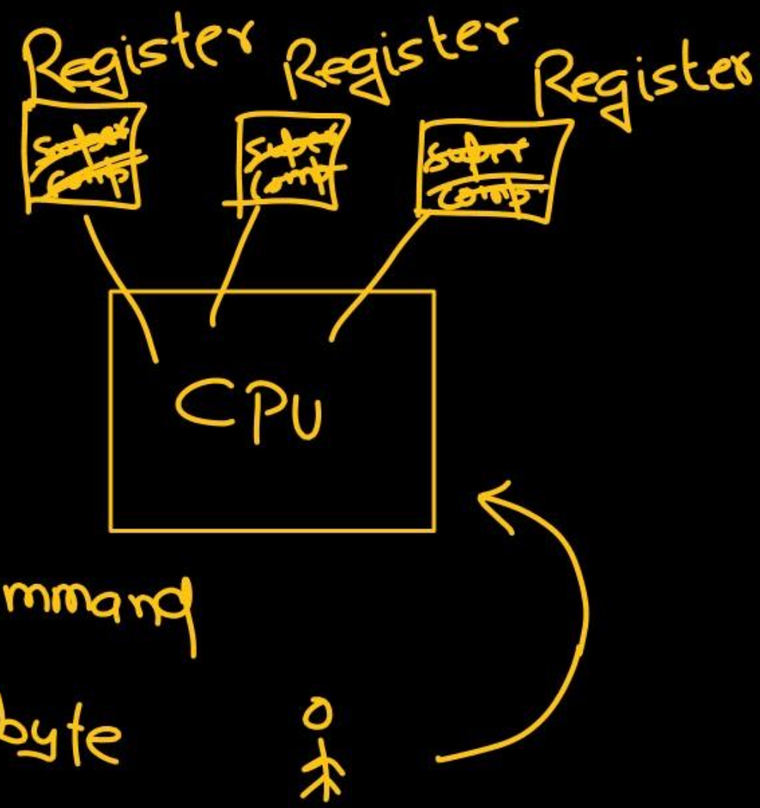
11 2


COA
some Register
brain
Processor
dynamic allocation
Heap Area
auto/ local
Stack
static, global variables
Uninitialized data segment
Initialized data segment
Code Section
static area

register

* As same as auto.
* storage : CPU register / stack





```
{
register int a;
}
```

reject ⇒

```
{
int a;
}
```

stack

grant ⇒

```
{
}
```

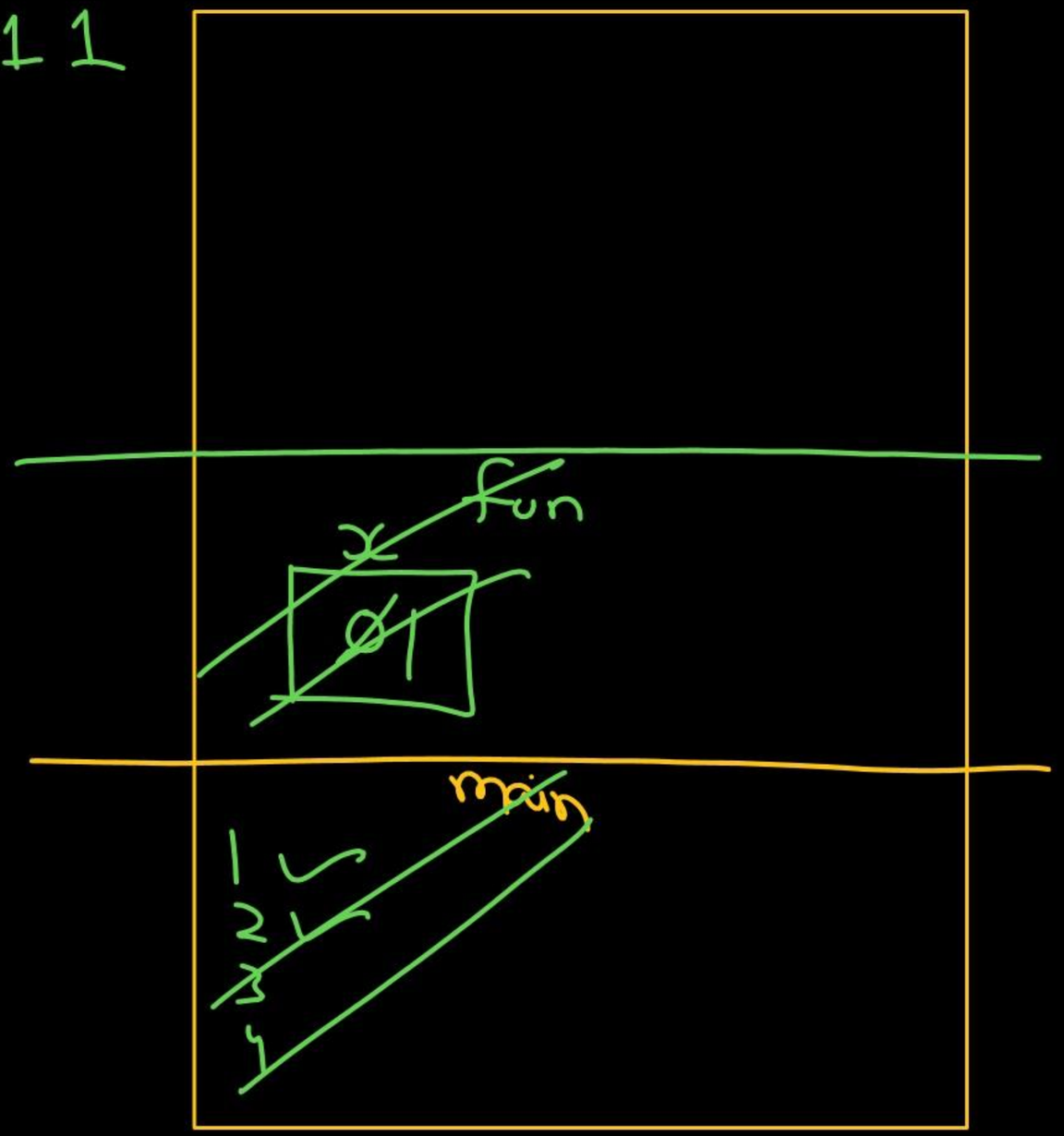
CPU register

CPU register/stack

111

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

void main(){
    1. fun();
    2. fun();
    3. fun();
    4. }
```



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.

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

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

1 2 3

data segment

3
~~0~~ ~~1~~ ~~2~~

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

b

Scope

① int a;
static int b = a;

int a=10, b=0;

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

Scope

a ~~1~~

b 1

① int a;
static int b = a;

X

"%d %d"

space

a 11 ~~10~~

b 1

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;
}

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

x

~~32~~ ~~01~~

13

Global → not inside a function ✓

dec/defined

→ default : 0 ✓

→ static area ✓

→ External

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

x?

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

forward decl

No Error

```
void f1(){
    extern int x;(No memory)
    ++x;
}
void f2(){
    extern int x; (No memory)
    ++x;
}
int x = 10;
void main(){
    ++x;
    f1();
    f2();
    printf("%d",x);
}
```

Error

```
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);
    }
```


THANK YOU GW
SOLDIERS !