unacademy

Functions and Storage Classes - Part II

Comprehensive Course on C- Programming

Pankaj Sharma • Lesson 14 • May 17, 2024

unacademy

CS & IT Engineering

C Programming

Functions & storage classes-II

Lecture - 14

By- Pankaj Sir

unacademy

Topics

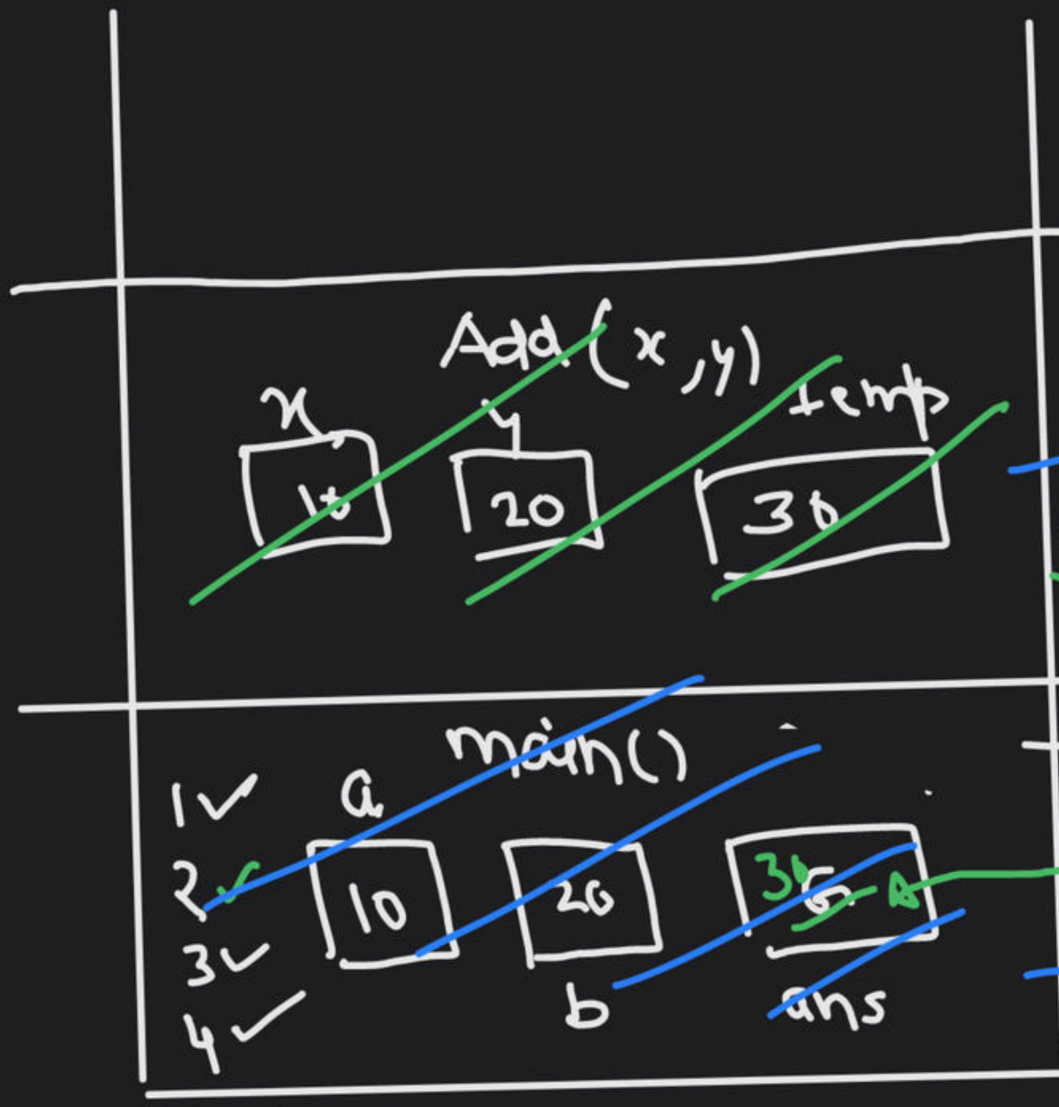
to be covered

1. Functions & storage class

unacademy

How function works

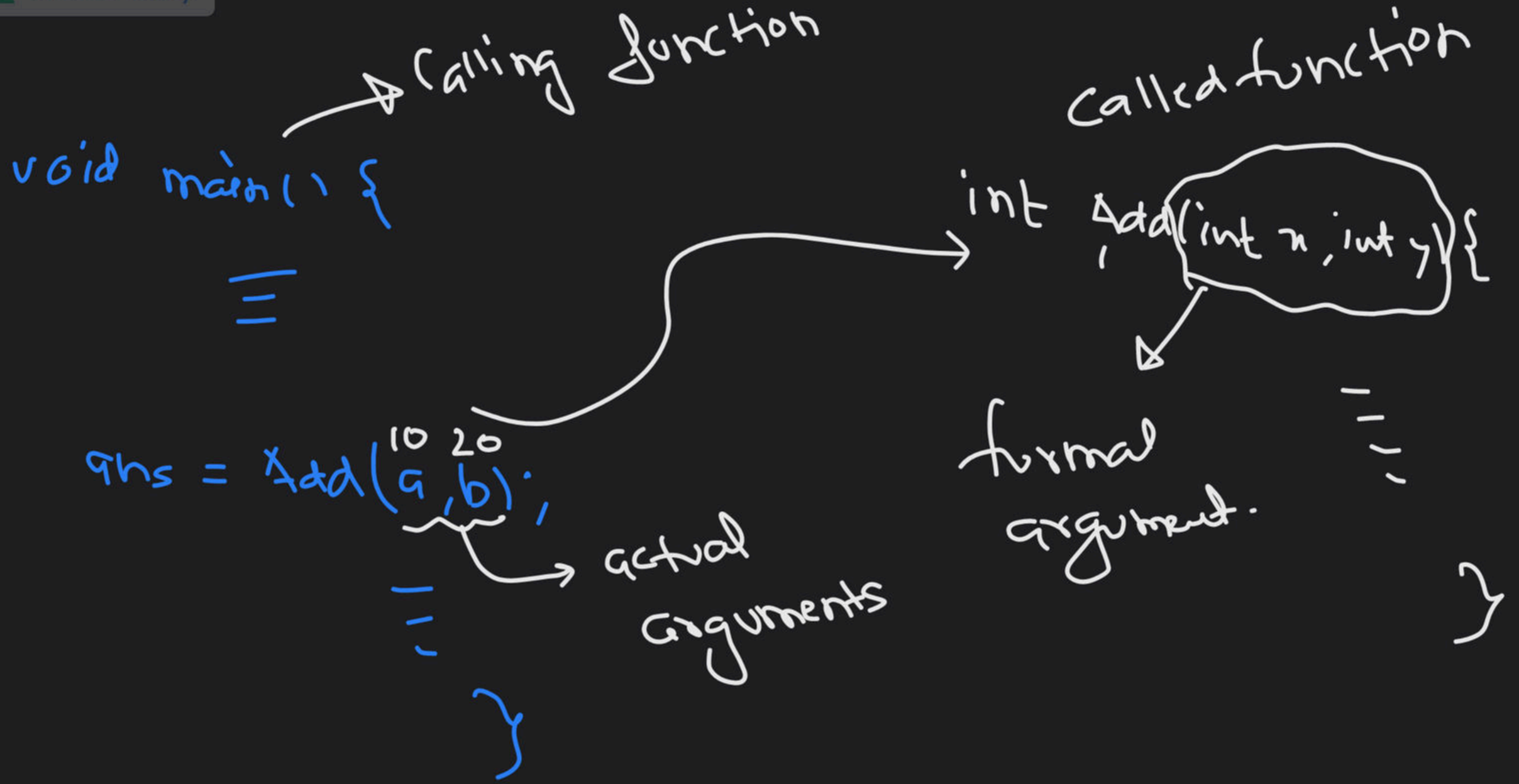
```
#include<stdio.h>
int Add(int,int);
void main(){
int a=10, b=20, ans;
ans = Add(a,b);
bf("%d", ans);
}
```




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

30

36


Calling function
Called function
void main() {
int Add(int x, int y) {
10 20
ans = Add(a, b);
actual arguments
formal argument.
}
}

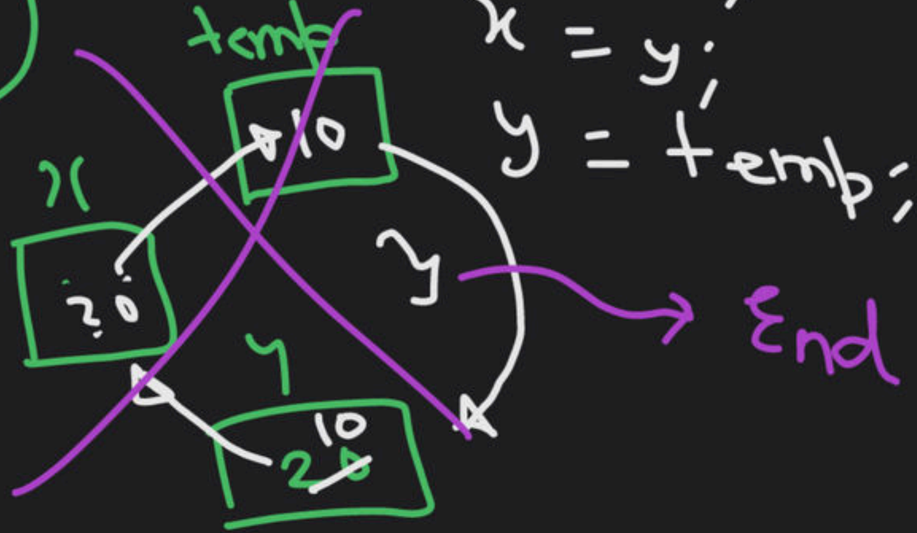
Swap

```
#include<stdio.h>
void swap(int,int);
void main(){
   int a=10,b=20;
printf("before swap");
printf("a=%d, b=%d",a,b);
      swap(a,b);
printf("a=%d, b=%d",a,b);
}
```

a → 10

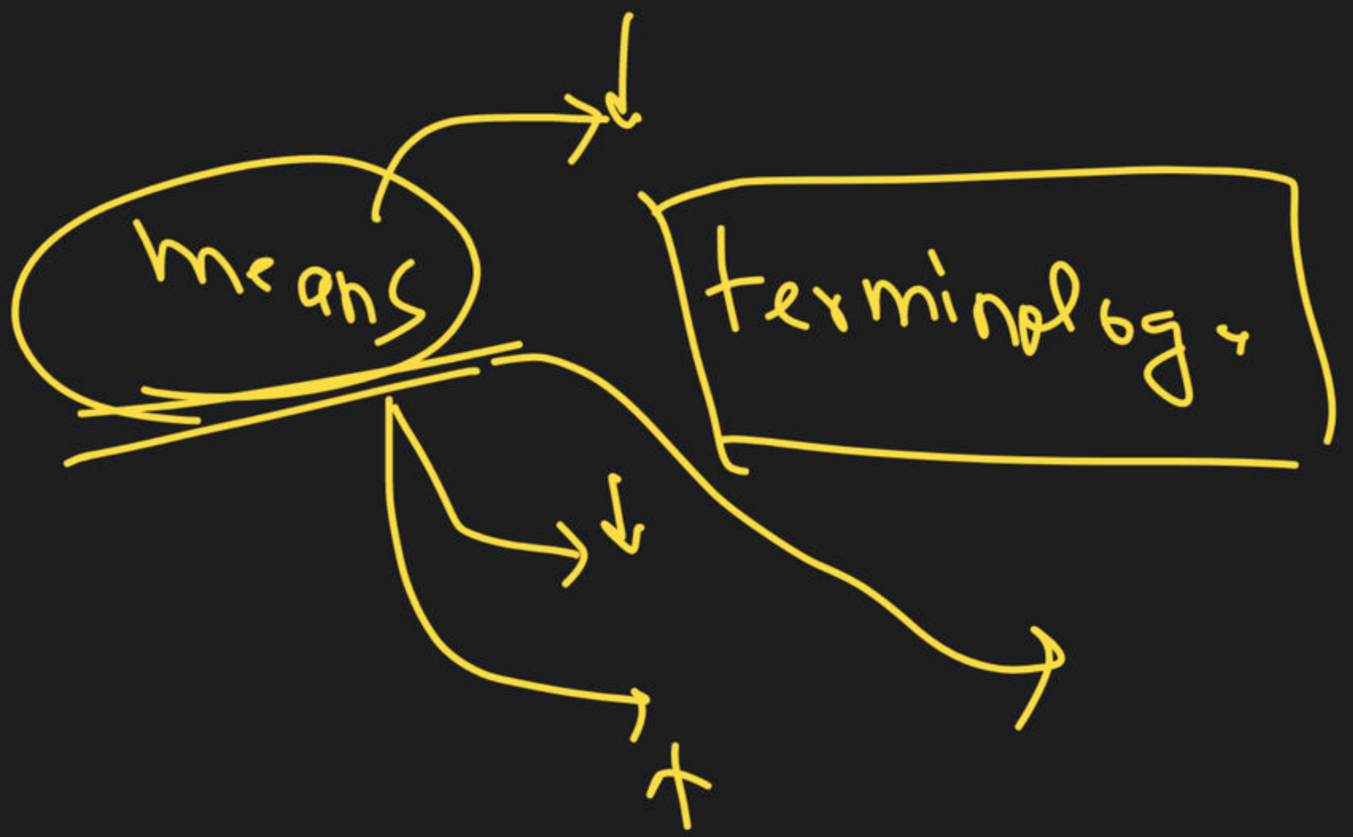
b → 20

```
void swap(int x, int y)
{
   int temp;
   temp = x;
   x = y;
   y = temp;
}
```



changes performed on formal arguments will not be reflected back to actual argument.

1. Call by value
2. call by reference / pass. — Address



Calling function

f ( )

f (10 / a,b)

actual argument

f (a,b)

int a

called func.

unacademy

Storage class

1. scope : part of code in which a variable is visible (where we can directly access the var.)
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 var. is stored.

```
void main(){
    int i;
    pf("%d", i);
}
```

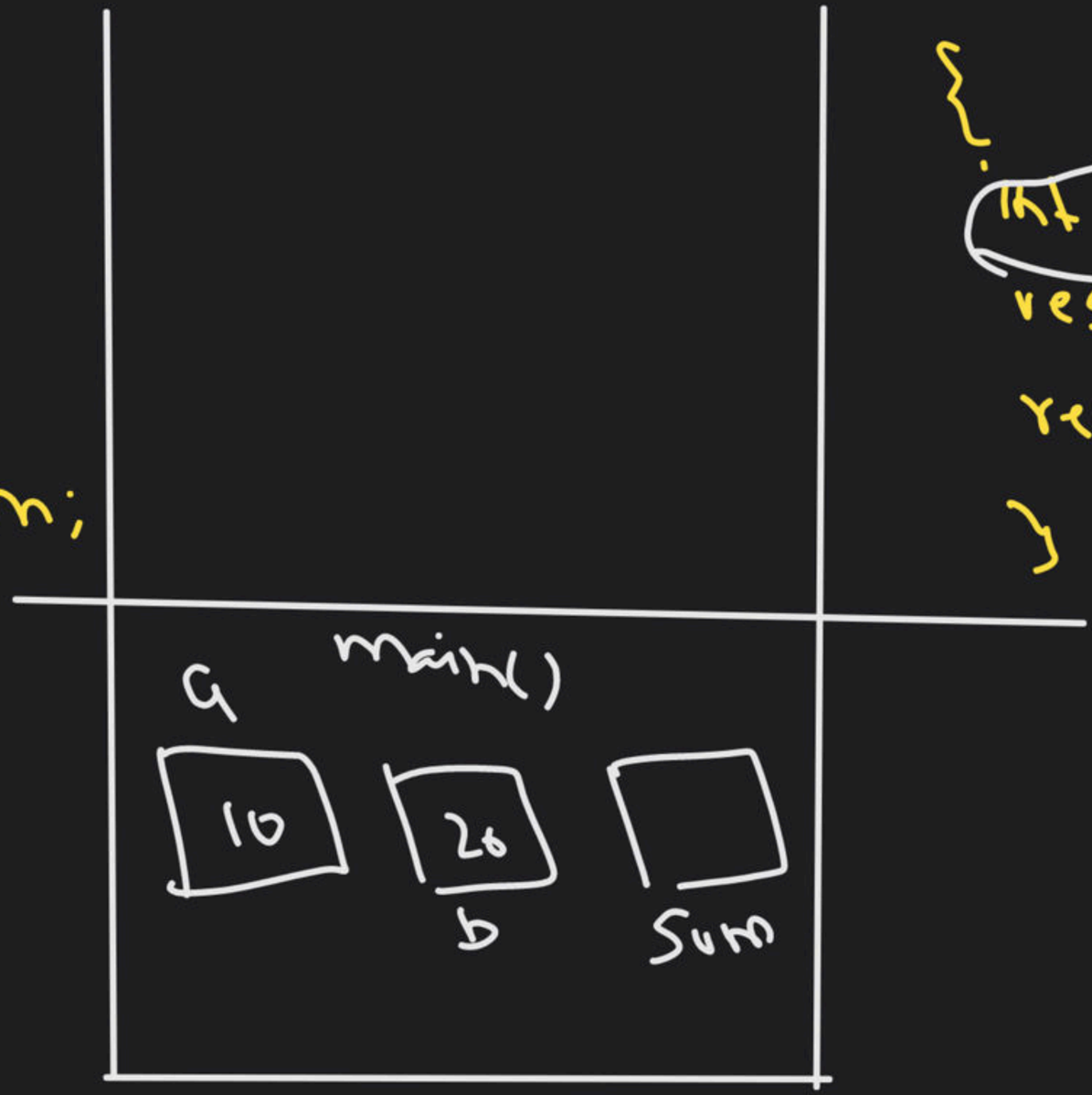
Garbage value

unacademy

① auto/local

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

void main() {
    int a=10,b=20;sum;
    sum = Add(a,b);
    pf("%d",sum);
}
```

```
int Add(int x,int y)
{
    int res;
    res = x+y;
    return res;
}
```



① block:

{

block

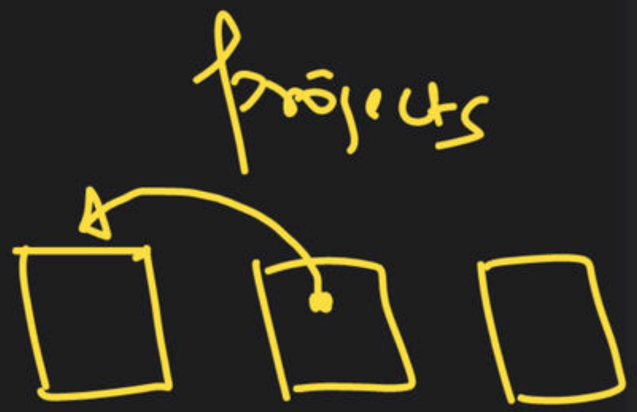
}

② File

```
void f1() {
    int a;
}
void f2() {
    int y;
}
```

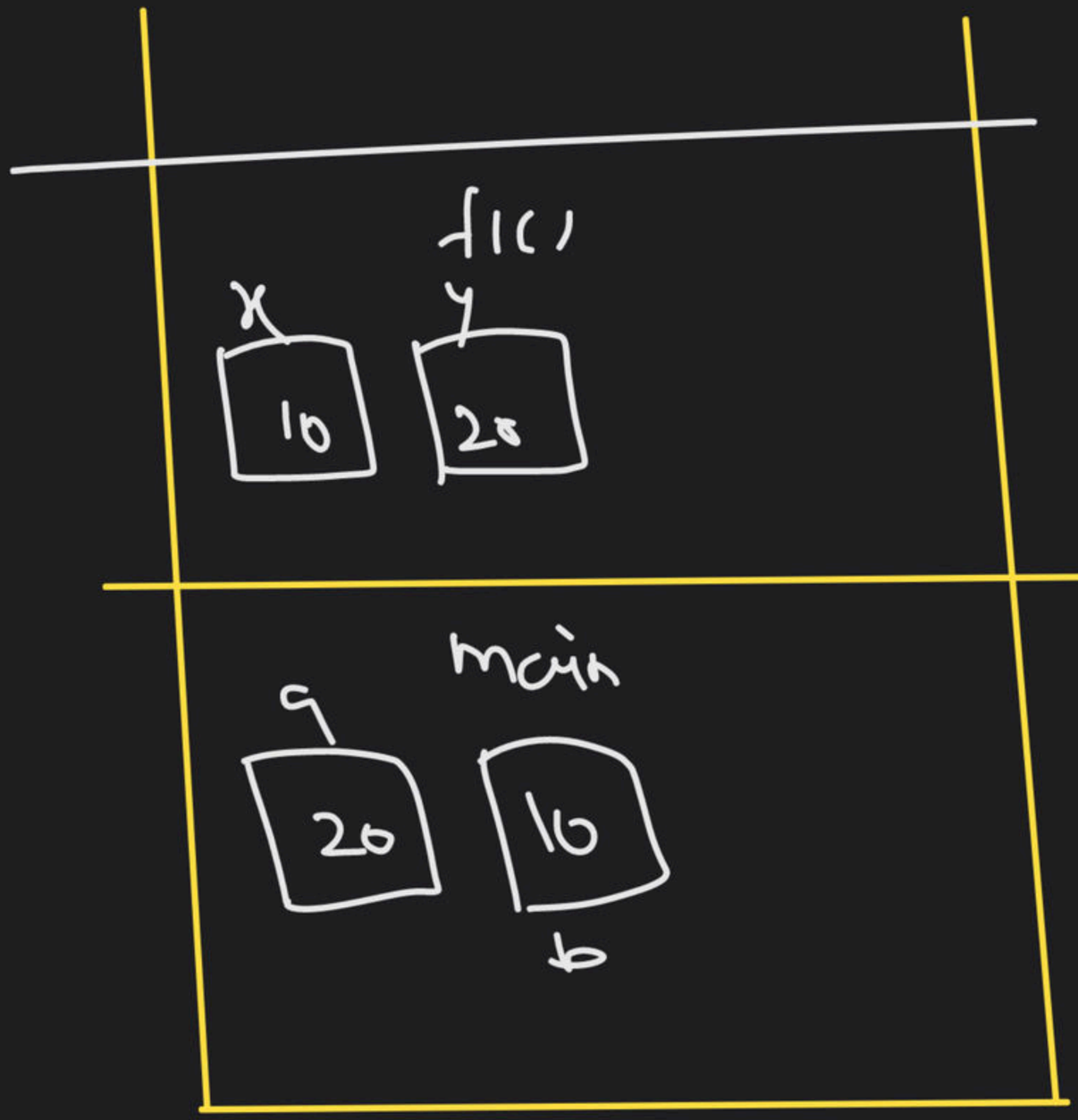
Pankaj.c

③ Multiple files



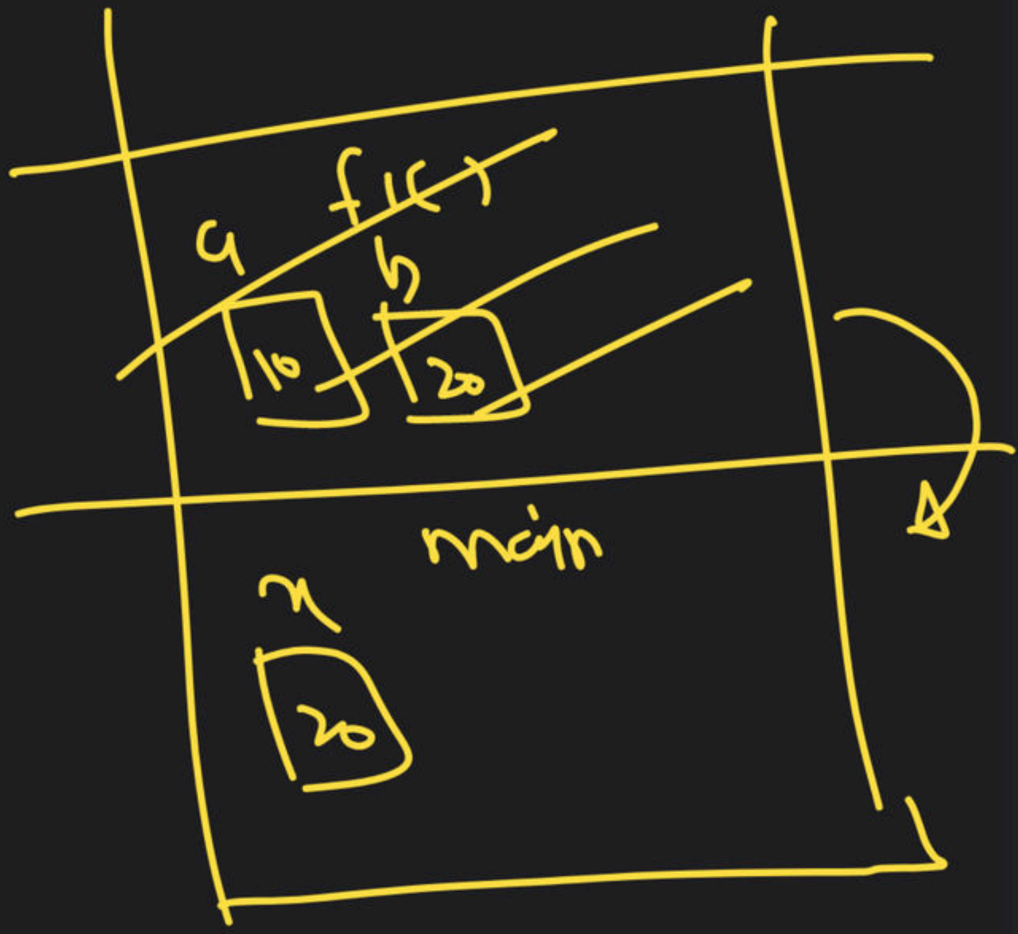
unacademy

```
void f1() {
    int x=10, y=20;
    pf("%d", x+y);
}

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



unacademy

```
void f1(){
    int a=10, b=20;
    pf("%d", a+b);
}

void main(){
    int x=20; f1()
    pf("%d", a+x);
}
```



unacademy

auto

By default variable defined/declared inside a function are auto.

```
void main(){
int a=10,b=20;
=
}
```

OR

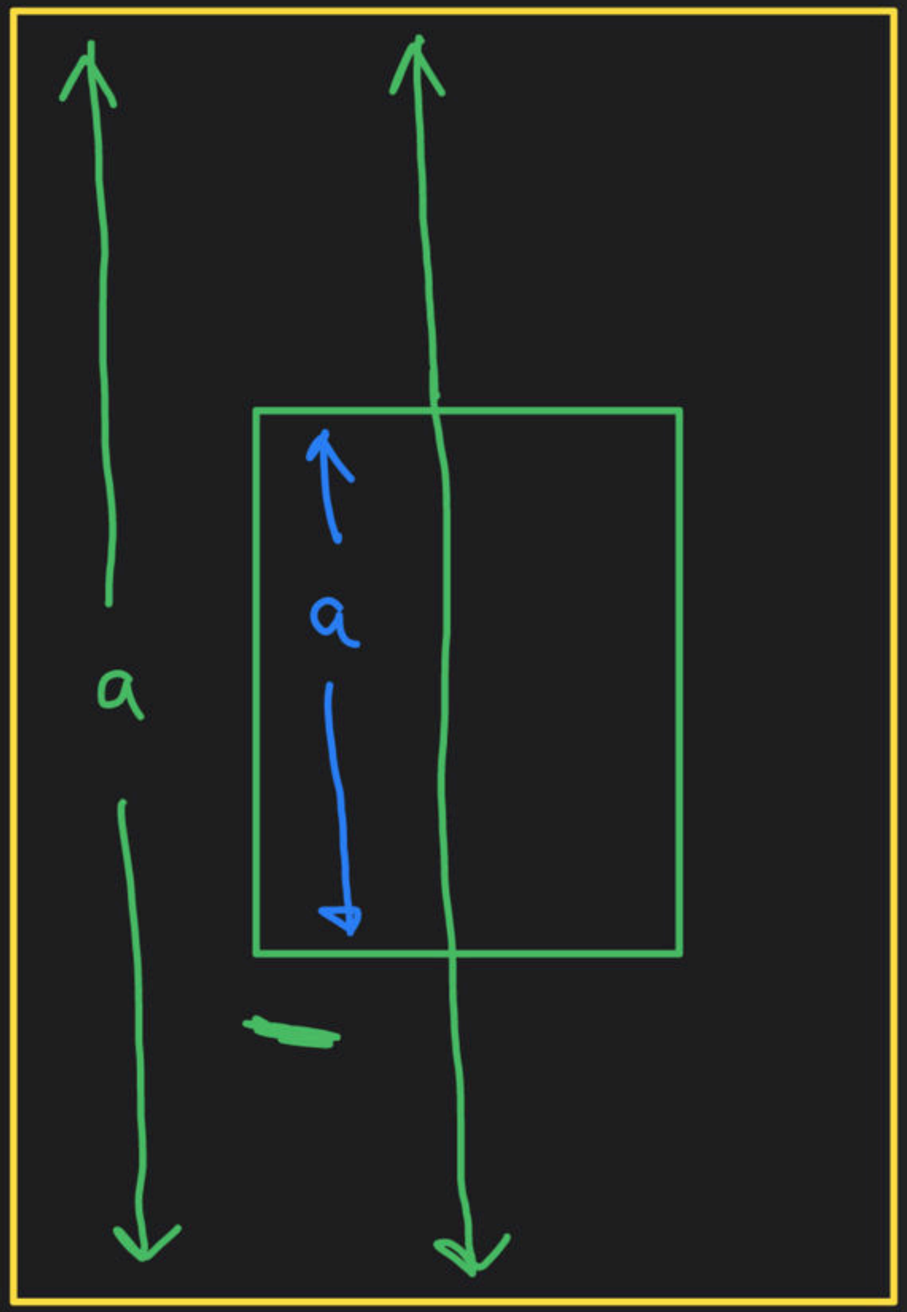
```
Void main(){
auto int a=10,b=20;
=
}
```

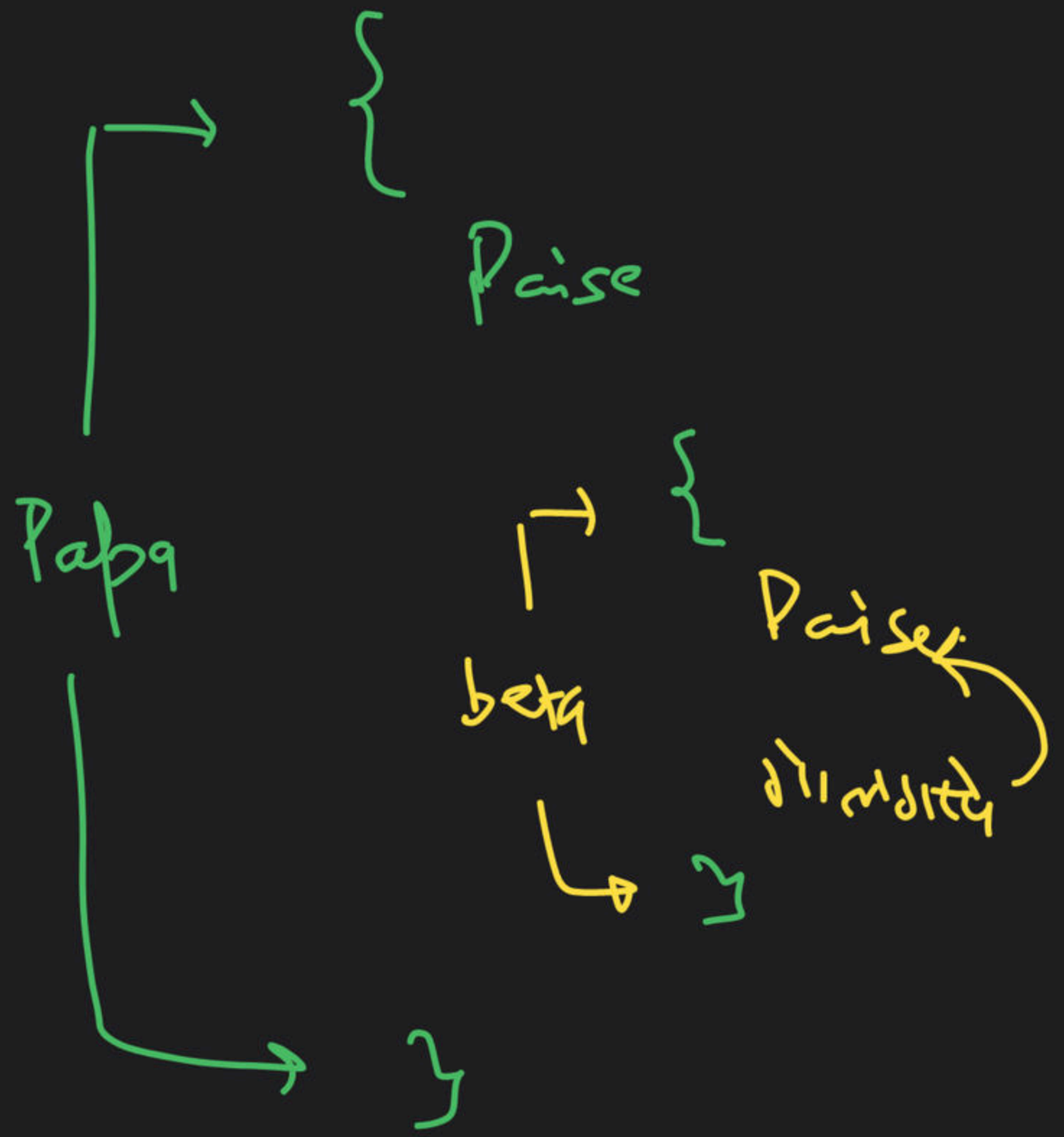
Auto

1. scope: block in which they are defined/declared
2. Lifetime: block in which they are declared.
3. default value: Garbage.
4. store : stack

21 12

```
void main(){
  int a=10;
  ++a;
  {
    int a=20;
    ++a;
    pf(".1.d",a);
  }
  ++a;
  pf(".1.d",a);
}
```





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

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

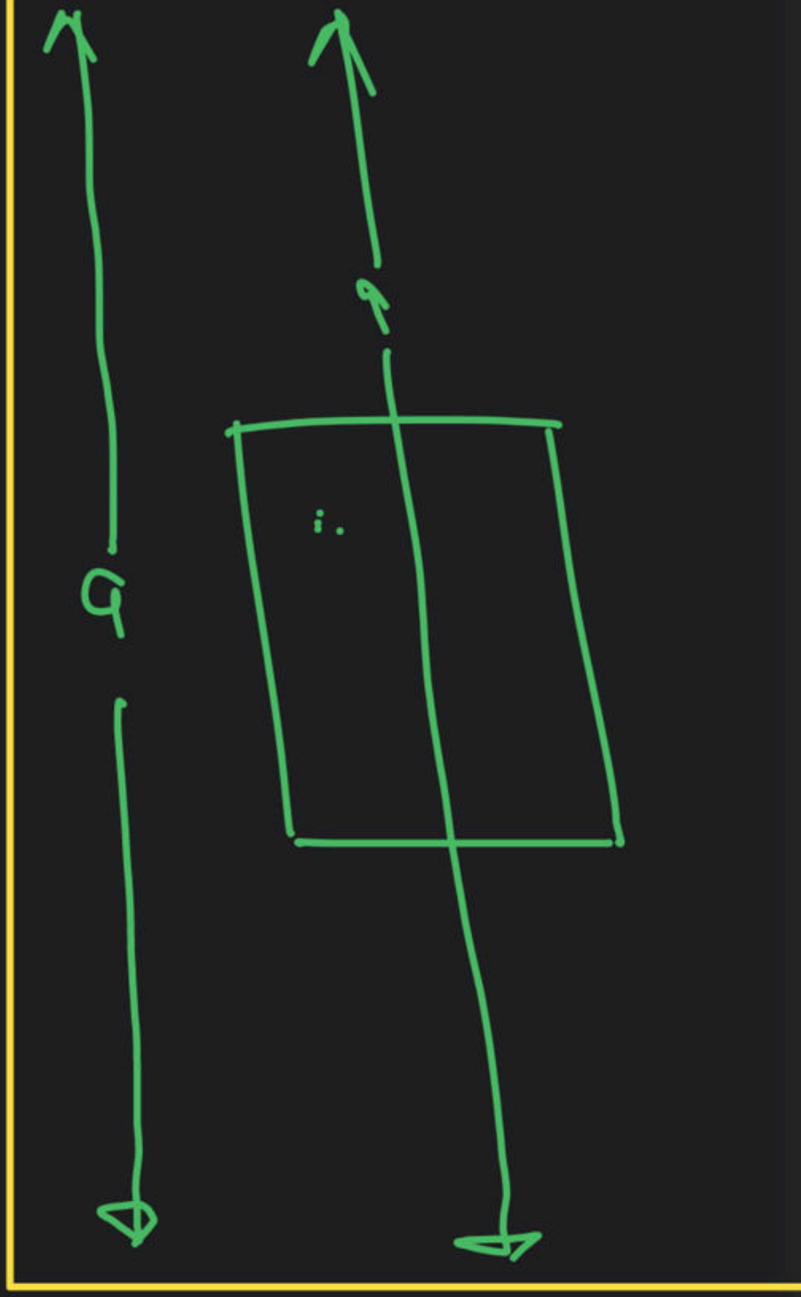
main

sub-scope

a

12 13

12 13



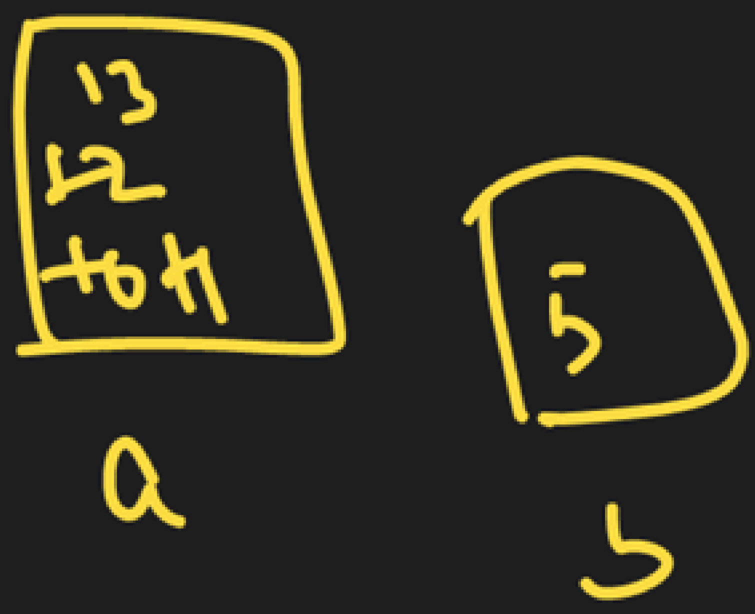
(1) Main scope variable is accessible within sub-scope.

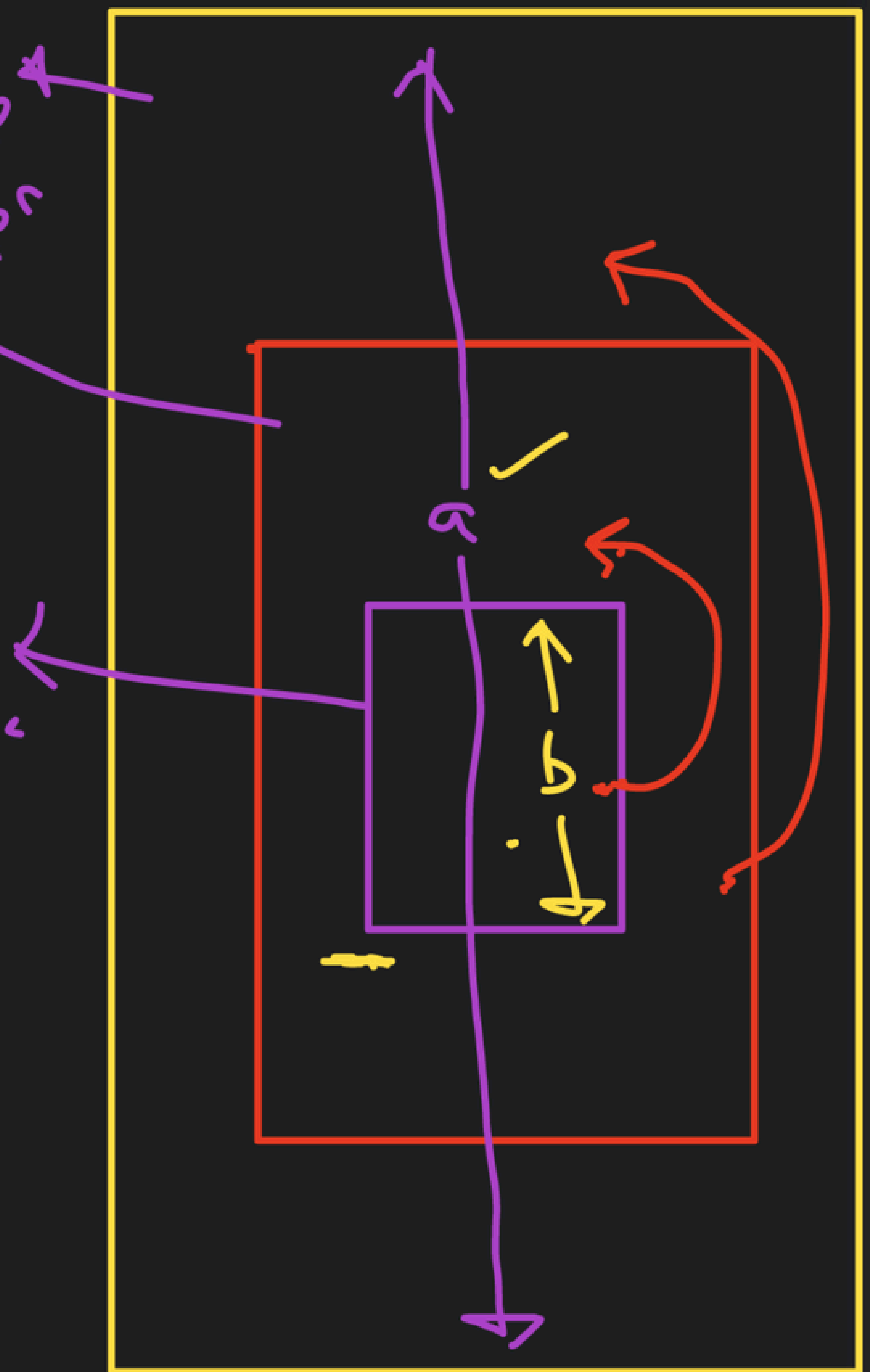
```
void main(){
 int a=10;
  ++a;
   {  ++a;
      pf("%d",a);
      { int b=5;
        ++a;
       pf("%d",a+b);
      }
    ++b;
    pf("%d %d",a,b);
   }
  ++a;
} pf("%d",a);
```

auto

b

va ke baat

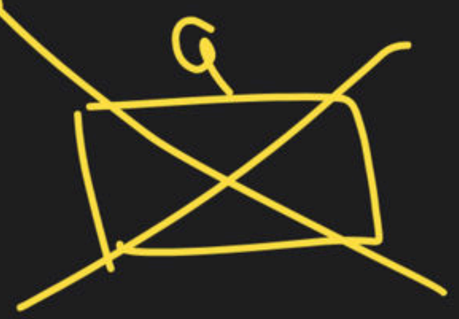




(2) Sub-scope variables are not accessible within Main scope.

auto ⇒ Created automatically when we enter the block in which they are declared and destructed automatically when we exit the block.

```
f() {
    int a;
}
```



unacademy

int a;

int a=10;

local/auto var.

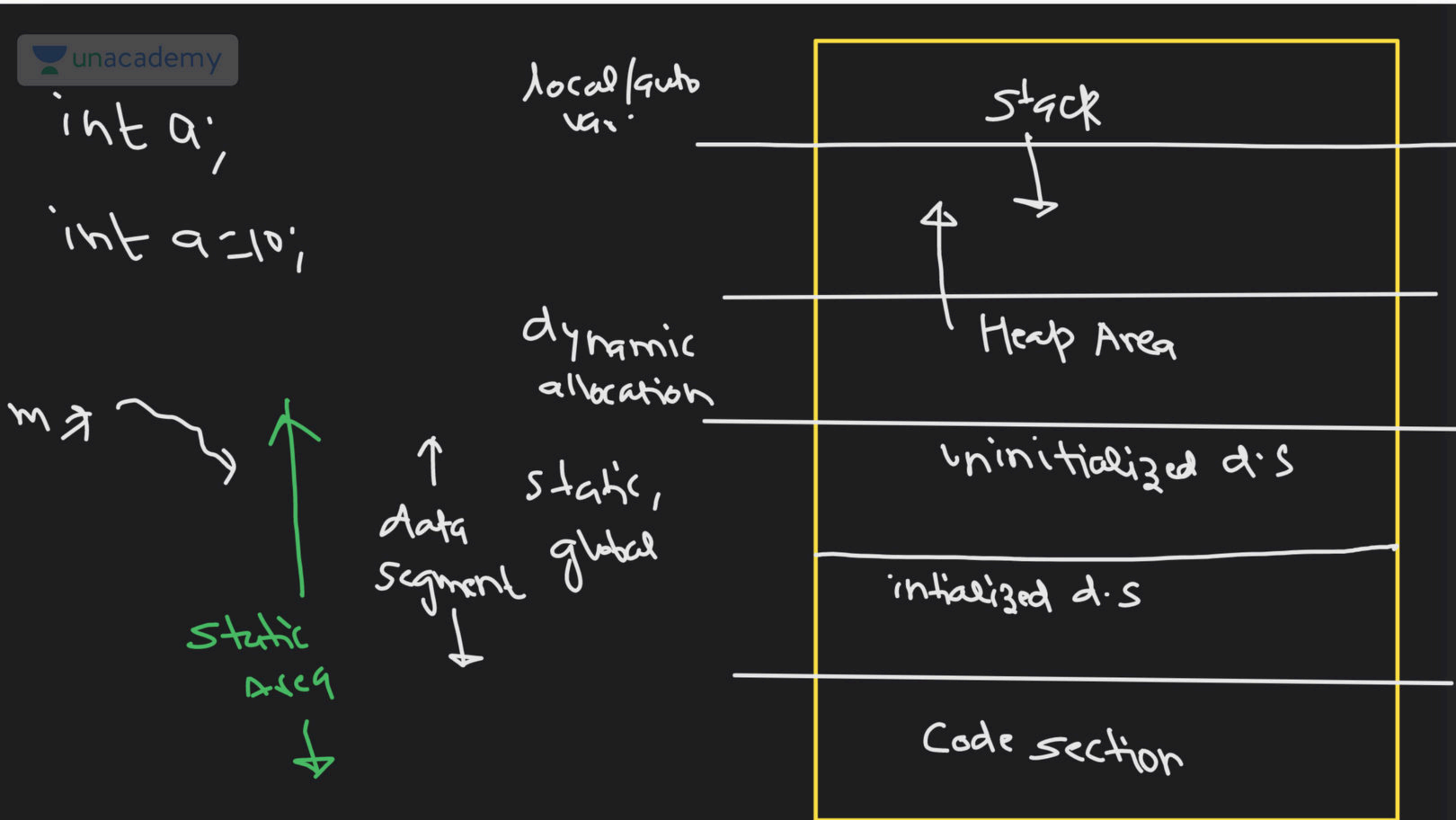
dynamic allocation

m

Data Segment

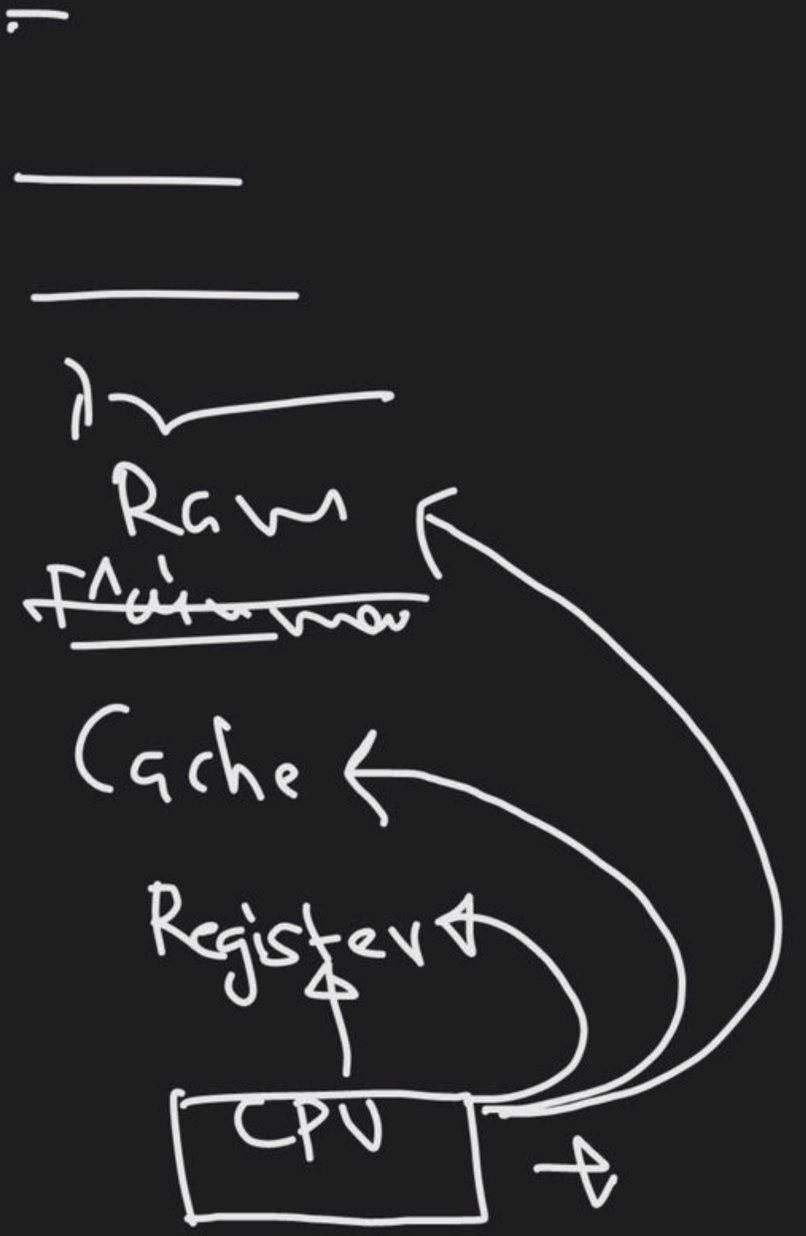
static, global

static area



Register

5th class student



unacademy

register variable

— Lifetime, scope, Default value

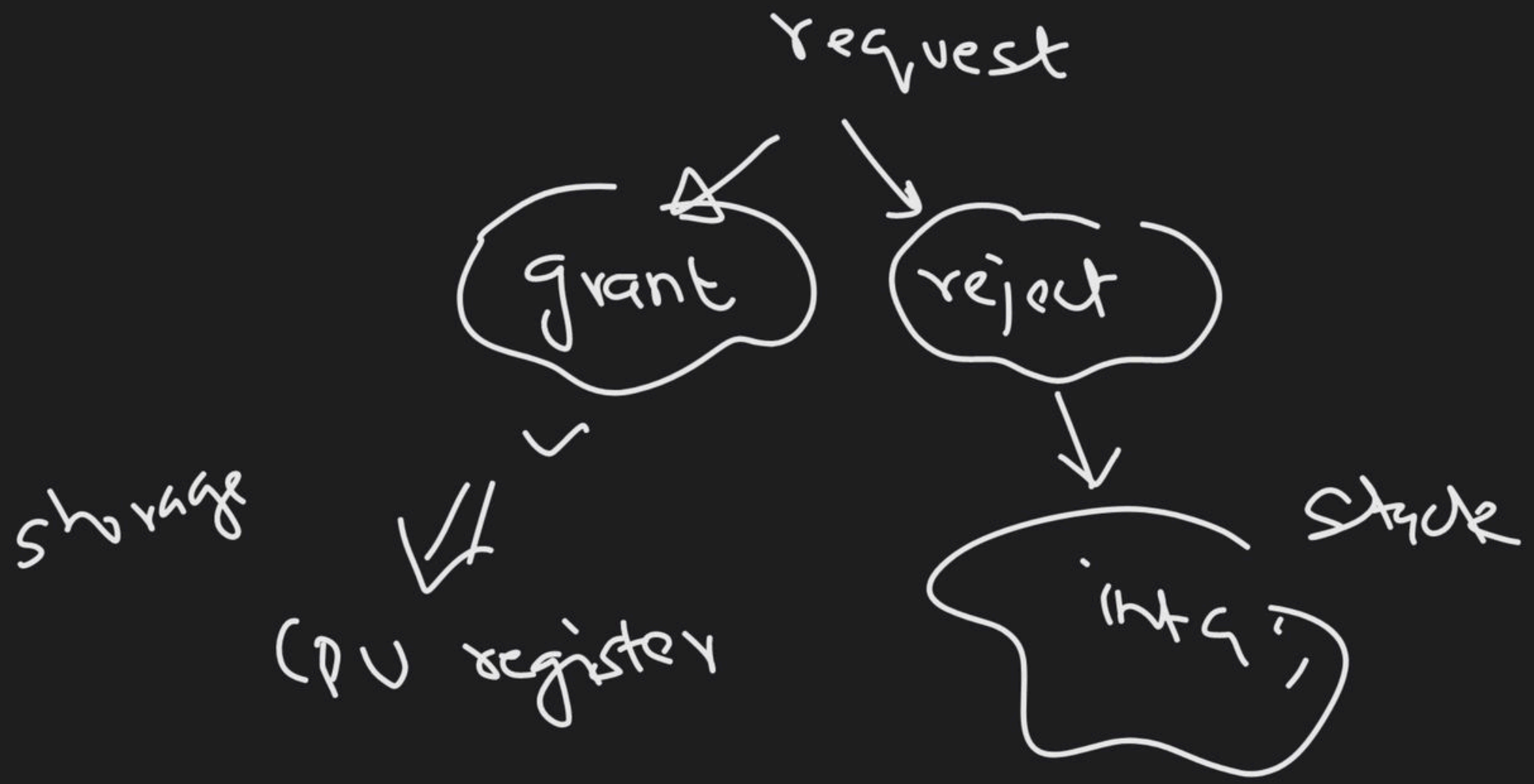
* As same as auto
* storage : CPU register | stack

void main(){

int a; → command

}

register int a;



(CPU register)
Diamond Inheritance

unacademy

DPP

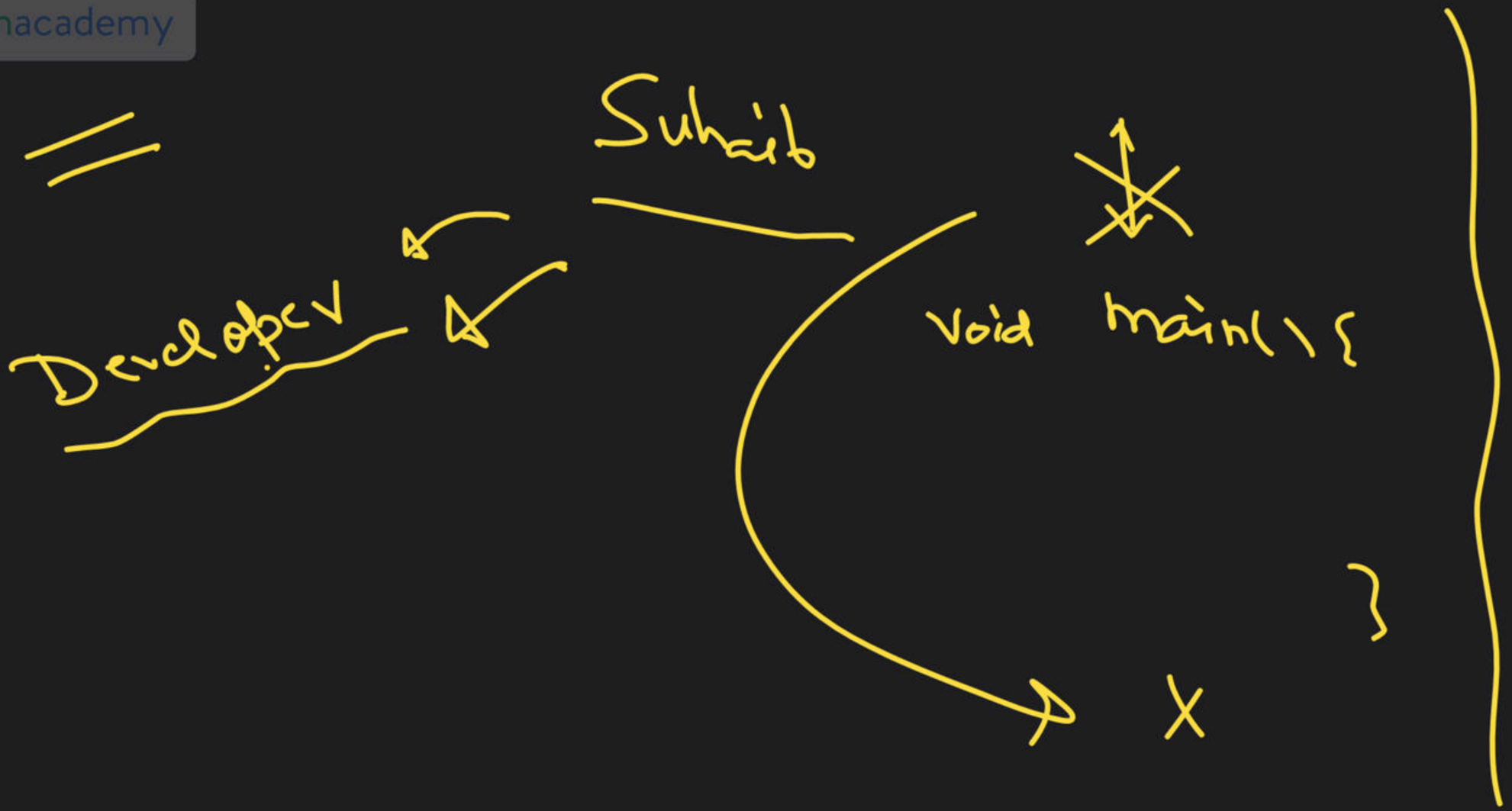
static
global variable
} 36 min

Recursion

Doubt ?

Agj

time Complexity



unacademy

THANK YOU!

Here's to a cracking journey ahead!