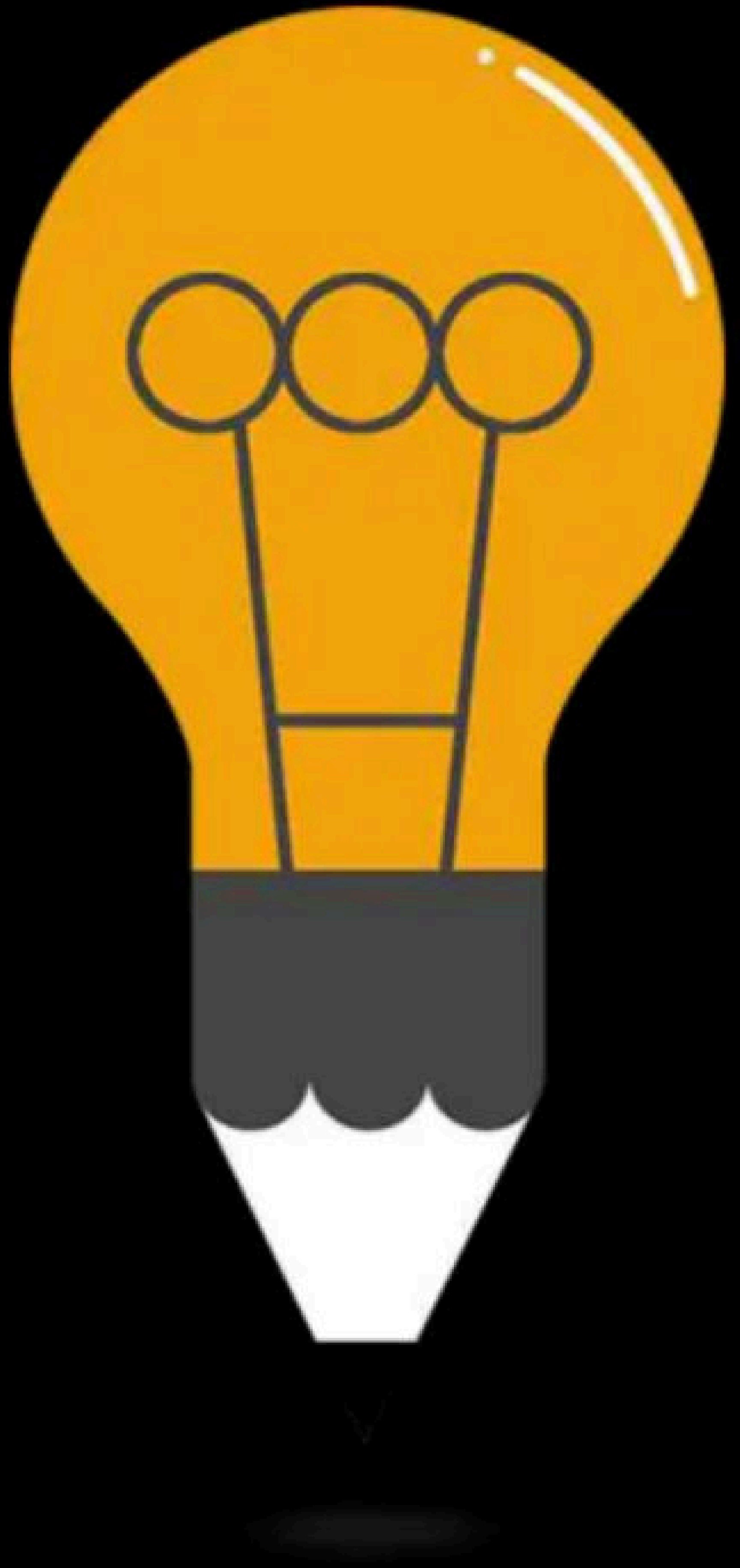
Constant, Macro, and Miscellaneous

Course on C-Programming & Data Structures: GATE - 2024 & 2025

Vishvadeep Gothi • Lesson 16 • Dec 14, 2022



Storage Classes

By: Vishvadeep Gothi

Dynamic memory allocation

↓

if any mem. locatⁿ assignment decided on run time.

---

malloc()

↳ malloc(size of memory in bytes)

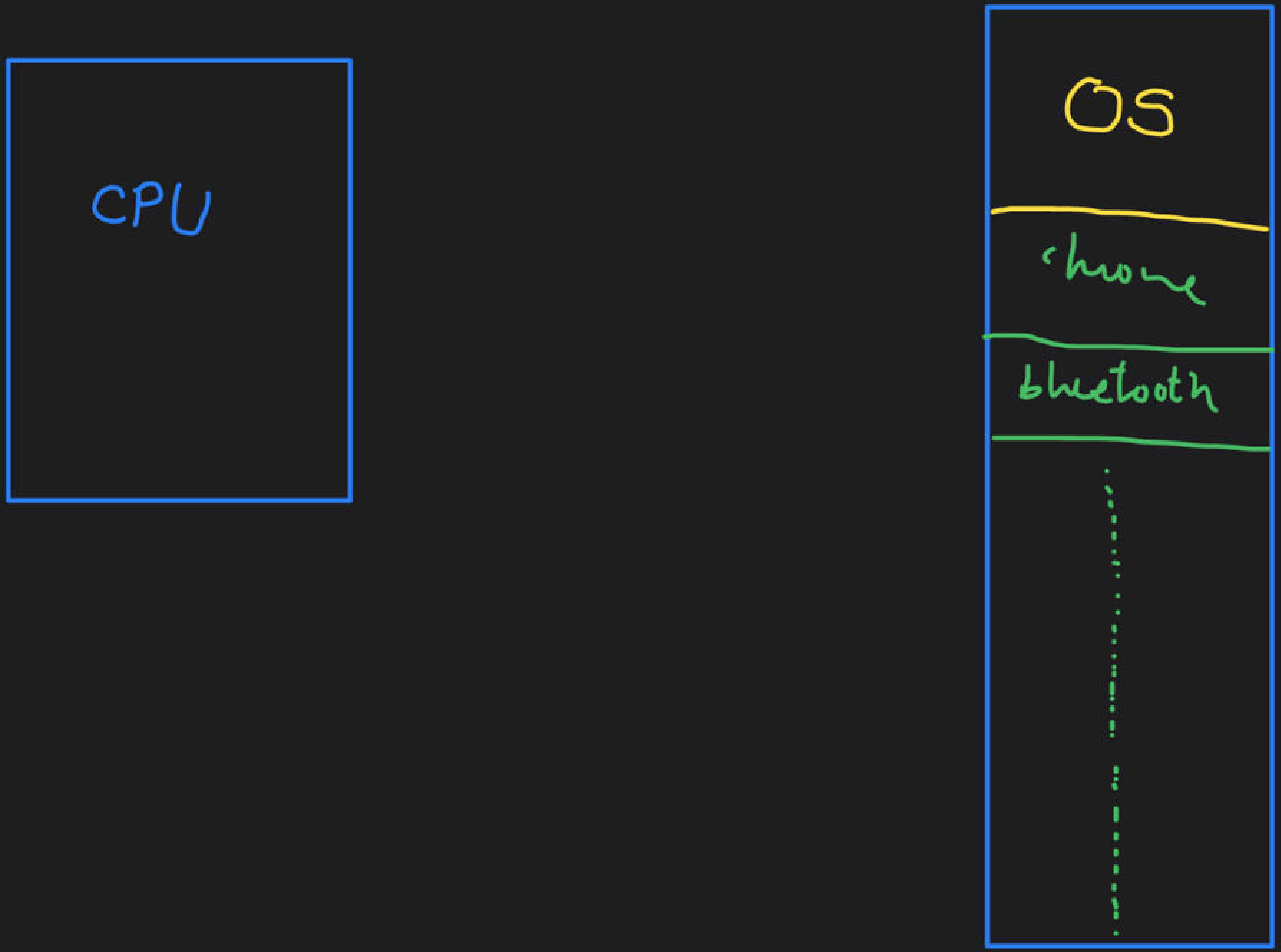
ex:-

`int x = (int)malloc(sizeof(int));` ×

malloc() returns address :-

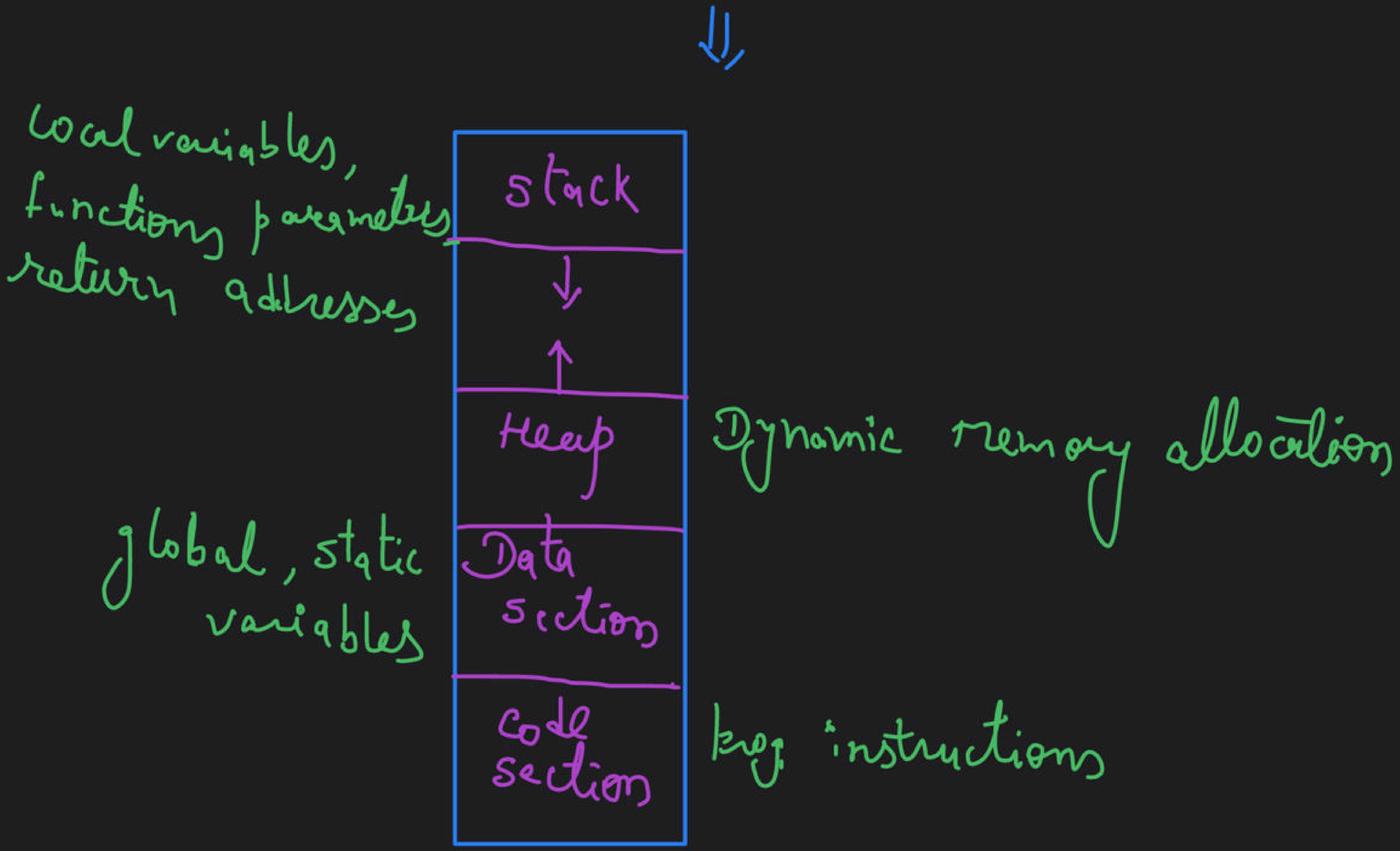
`int *p = (int *)malloc(sizeof(int));` ✓

```
void main()
{
  char choice;
  printf("you want to create a vaiable? —> y for yes");
  scanf("%c", &choice);
  if( choice=='y' || choice = = 'y')
  {
    int *p = (int *) malloc(sijeof(int));
    *p = 5;
    printf("%d", *p);
  }
```



all running prog.s and their content should be stored in RAM.

when a prog. is stored in RAM => it is stored in 4 sections

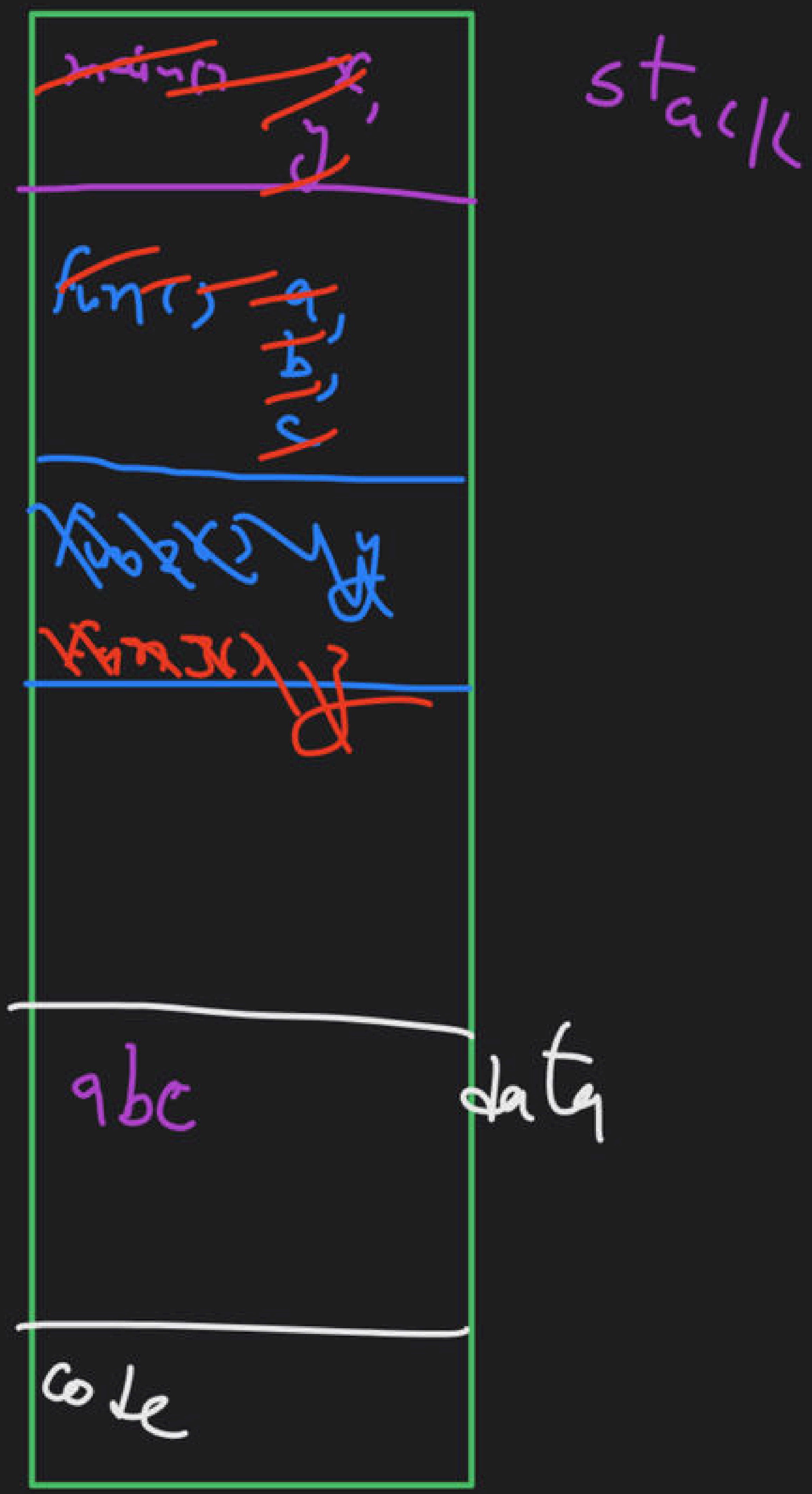


```
int abc;
void main()
{
 int x;
 char y;

fun(x, y);
printf("main over");
}
```

```
void fun(int a, int b)
{
  int c;
  fun2();
  printf(" fun2 call");
 fun3();
 printf(" fun1 over");
}
```

```
fun2()
{
char y;
  =
  =
  ;
}
```

```
fun3()
{
 float z;
} =
```

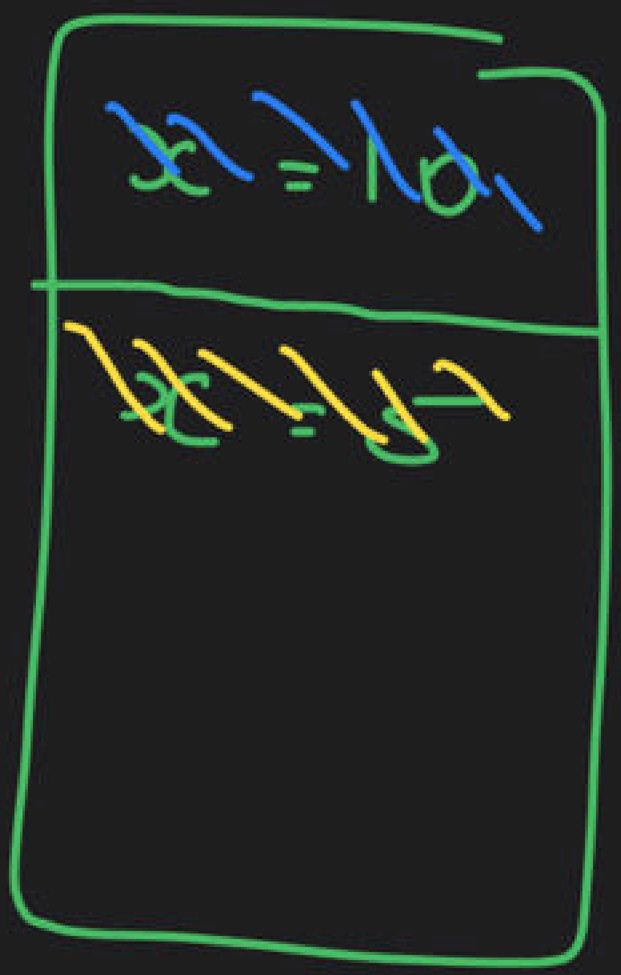

~~main() x, y~~
stack
~~fun() a, b, c~~
~~fun2() y~~
~~fun3() y~~
abc
data
code

blocks:-

```
{


}
```

```
void main()
{
  int x = 10;
  {
    int x = 5;
    printf("%d", x);
  }
  printf("%d", x);
}
```

stack



output :- 5 10

Characteristics of Variables

1. Lifetime
2. Scope
3. Initialization
4. Location

Storage Classes

1. auto
2. register
3. static
4. extern

auto

Local Variables

```
void main()
{
        int x;          or          auto int x;
```

auto

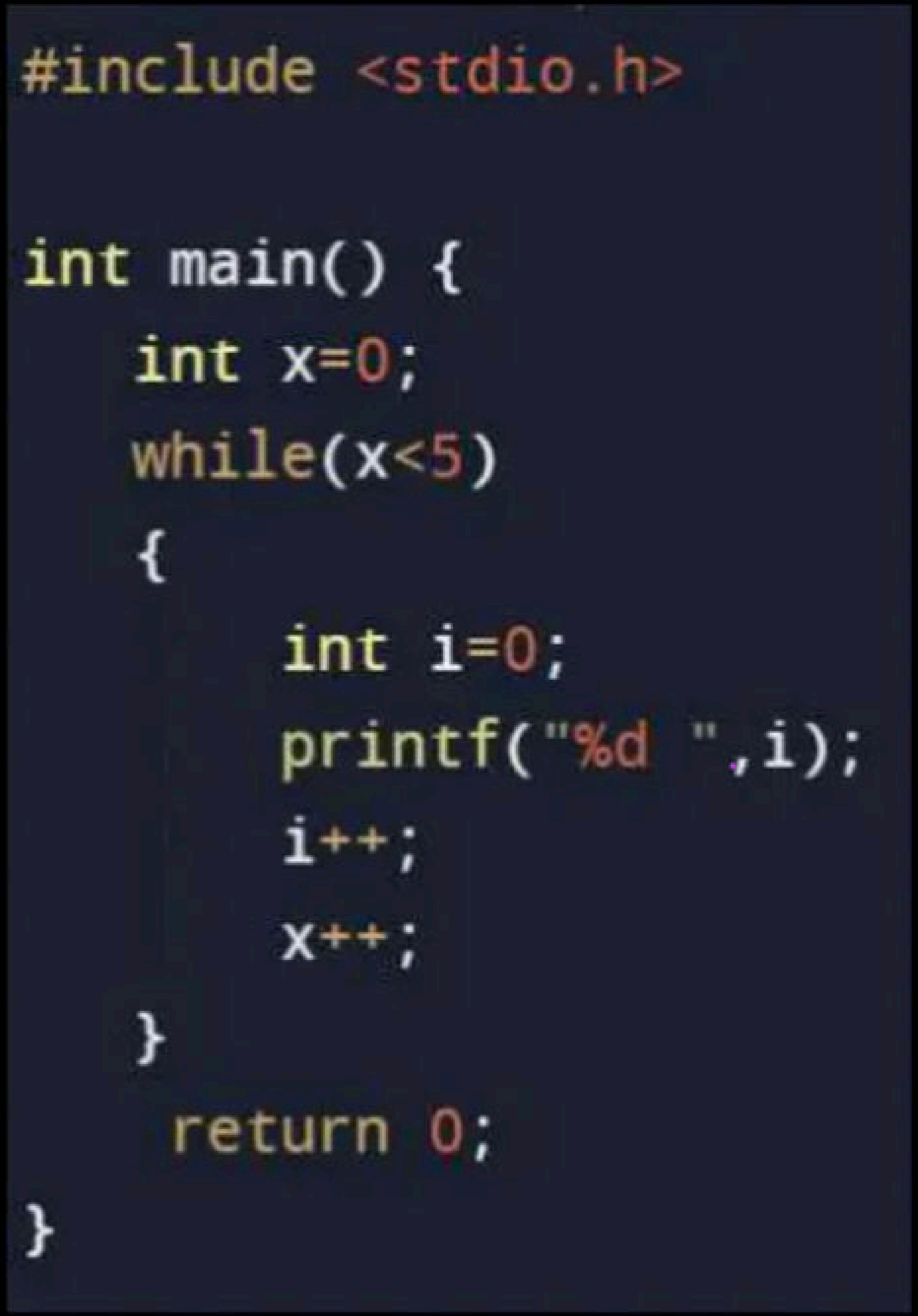
Lifetime => during funct^n execution or during block execution

Scope => within a function or within block

Initialization => default initializat^n with garbage value

Location => stack

auto

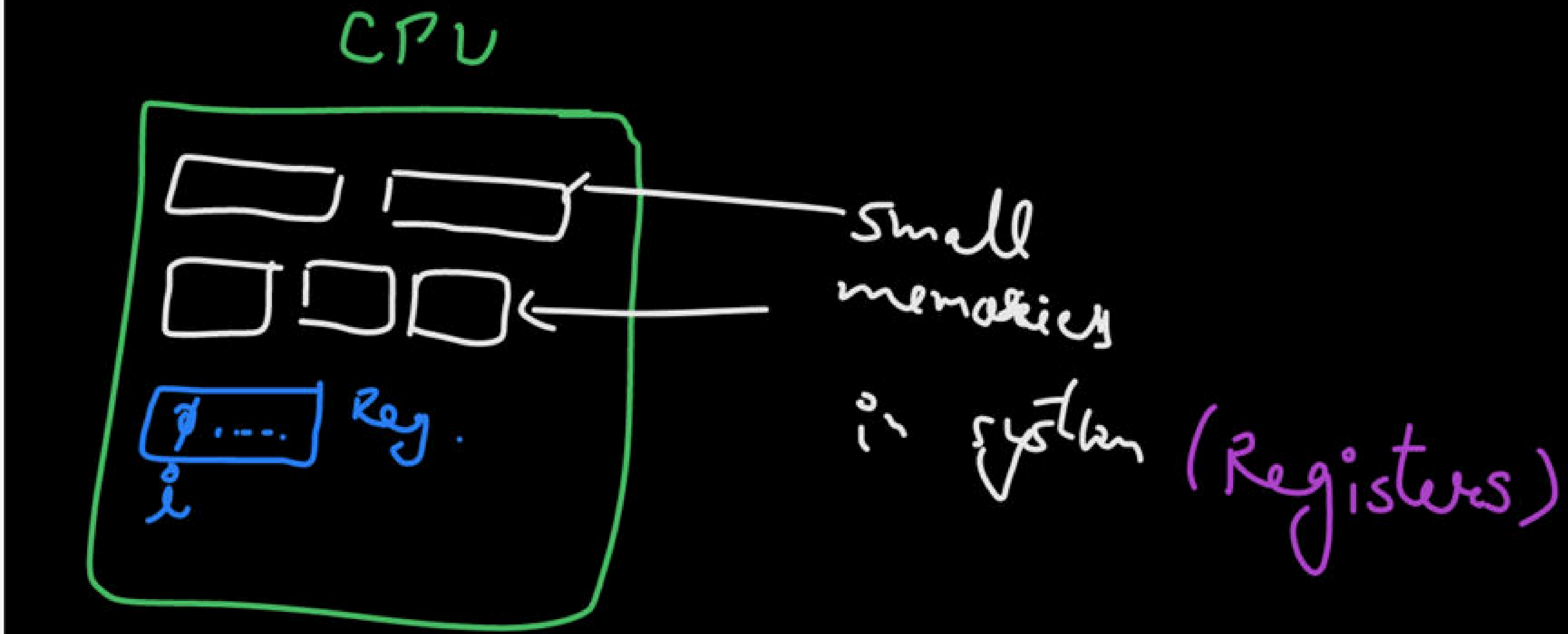


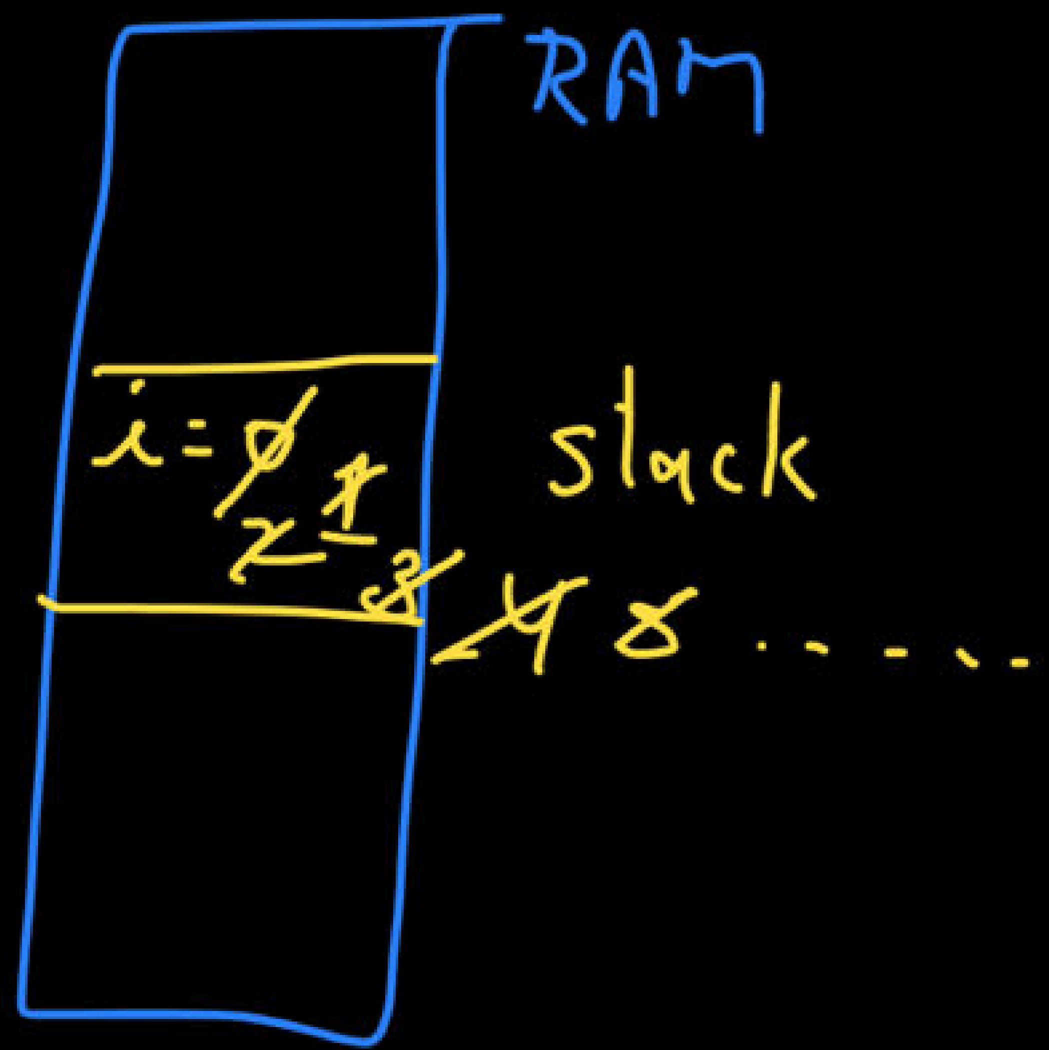
```
#include <stdio.h>

int main() {
   int x=0;
   while(x<5)
   {
       int i=0;
       printf("%d ",i);
       i++;
       x++;
   }
    return 0;
}
```

0 0 0 0 0

register

Local Variables and exactly like auto but storage is not in RAM (stack) but in CPU register





```
register int i;
    for(i = 0; i<1000; i++)
    {
      =
     }
```

Lifetime :- within funct^n or block

scope :- — " —— " ——

Initializat^n :- garbage

locat^n :- CPU reg. but if registers are not available then in stack in RAM.

Automatic Variables

1. auto
2. register

global variable :-

lifeline :- during prog. executⁿ

scope :- from entire prog.

initializatⁿ :- by default initialized with 0

locatⁿ :- data section

```
int x;
 void main()
{
 printf("%d", x);
}
```

output ⇒ 0

static

Local or global

```
static int x;

void main()
{
  static int y;

     -
   -
   -
   -
}
```

```
fun()
{

  printf("%d", y);

}
```

static

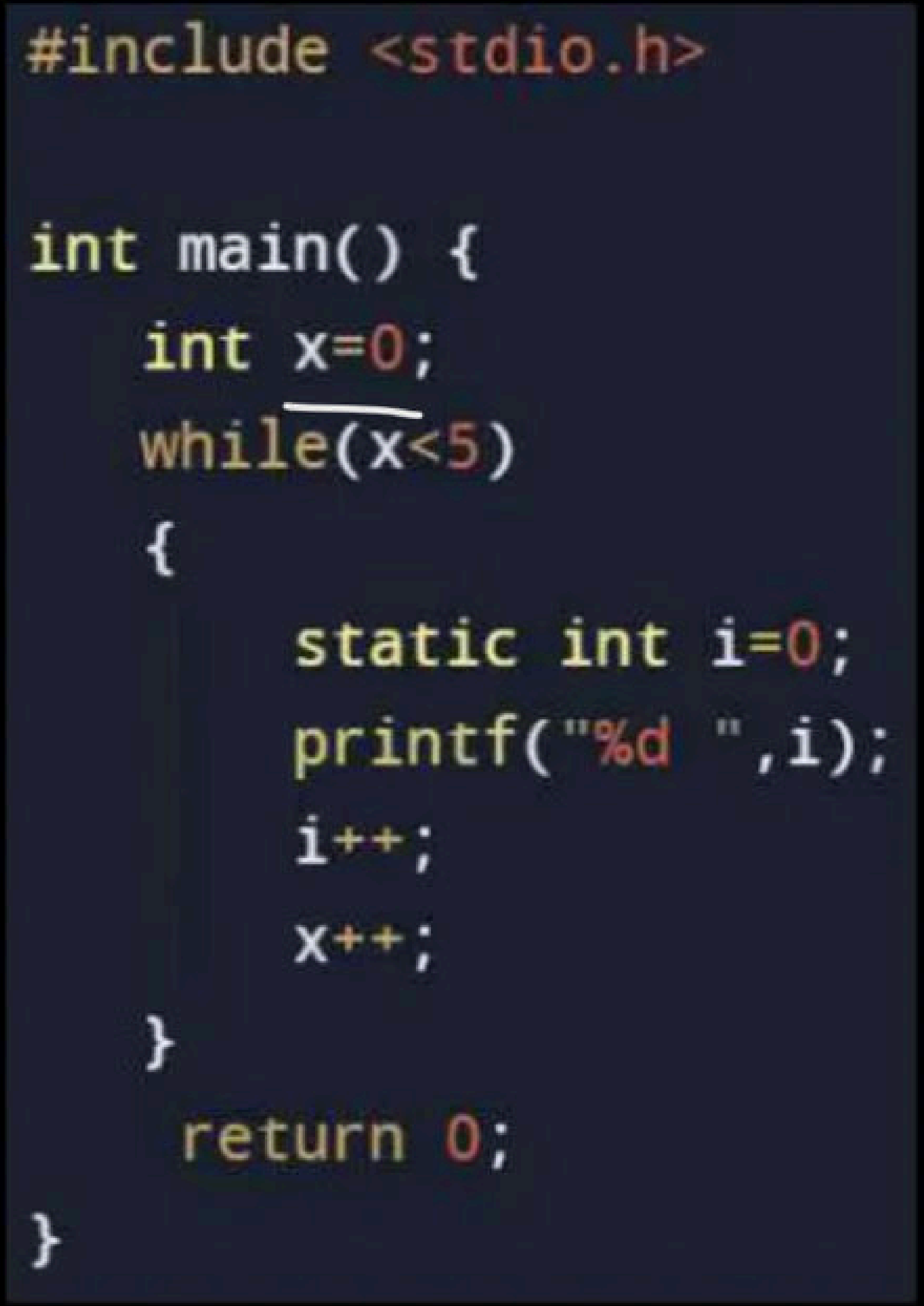
Lifetime ⇒ During program execution

Scope ⇒ local are global

Initialization ⇒ with zero

Location ⇒ data section

static



```
#include <stdio.h>

int main() {
   int x=0;
   while(x<5)
   {
       static int i=0;
       printf("%d ",i);
       i++;
       x++;
   }
    return 0;
}
```

x = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

i = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

0 1 2 3 4

extern

global

extern

Lifetime :- prog.

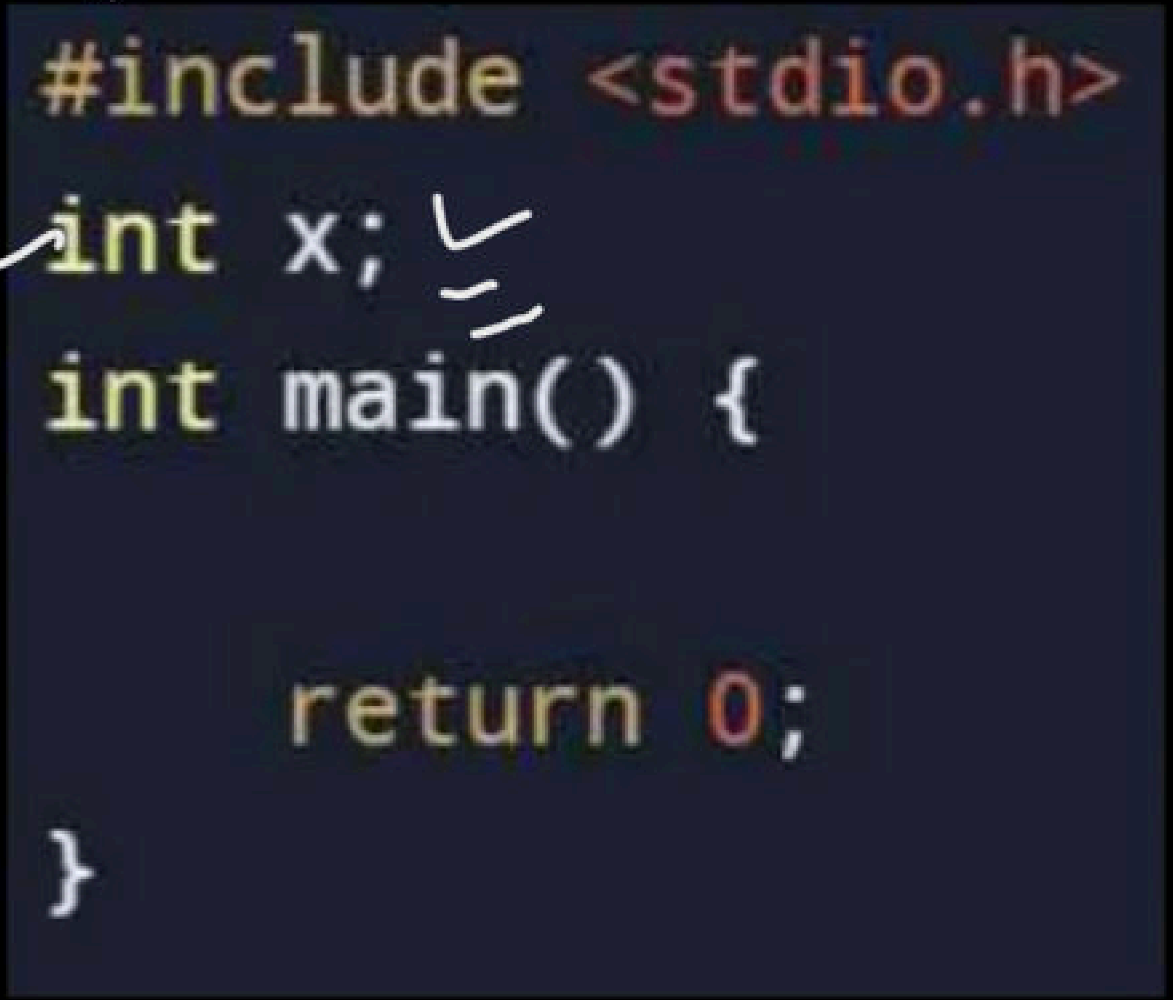
Scope :- prog.

Initialization :- with 0

Location :- data

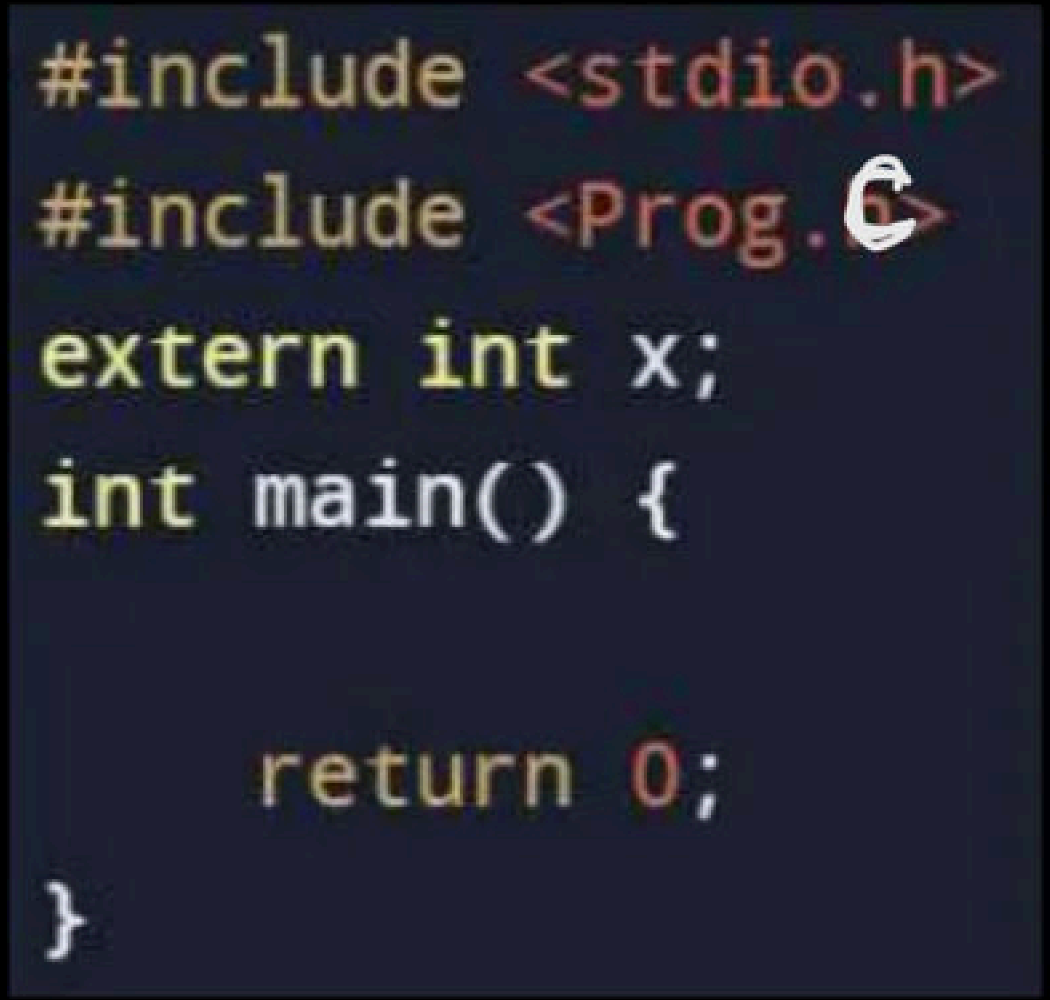
extern

Prog.c



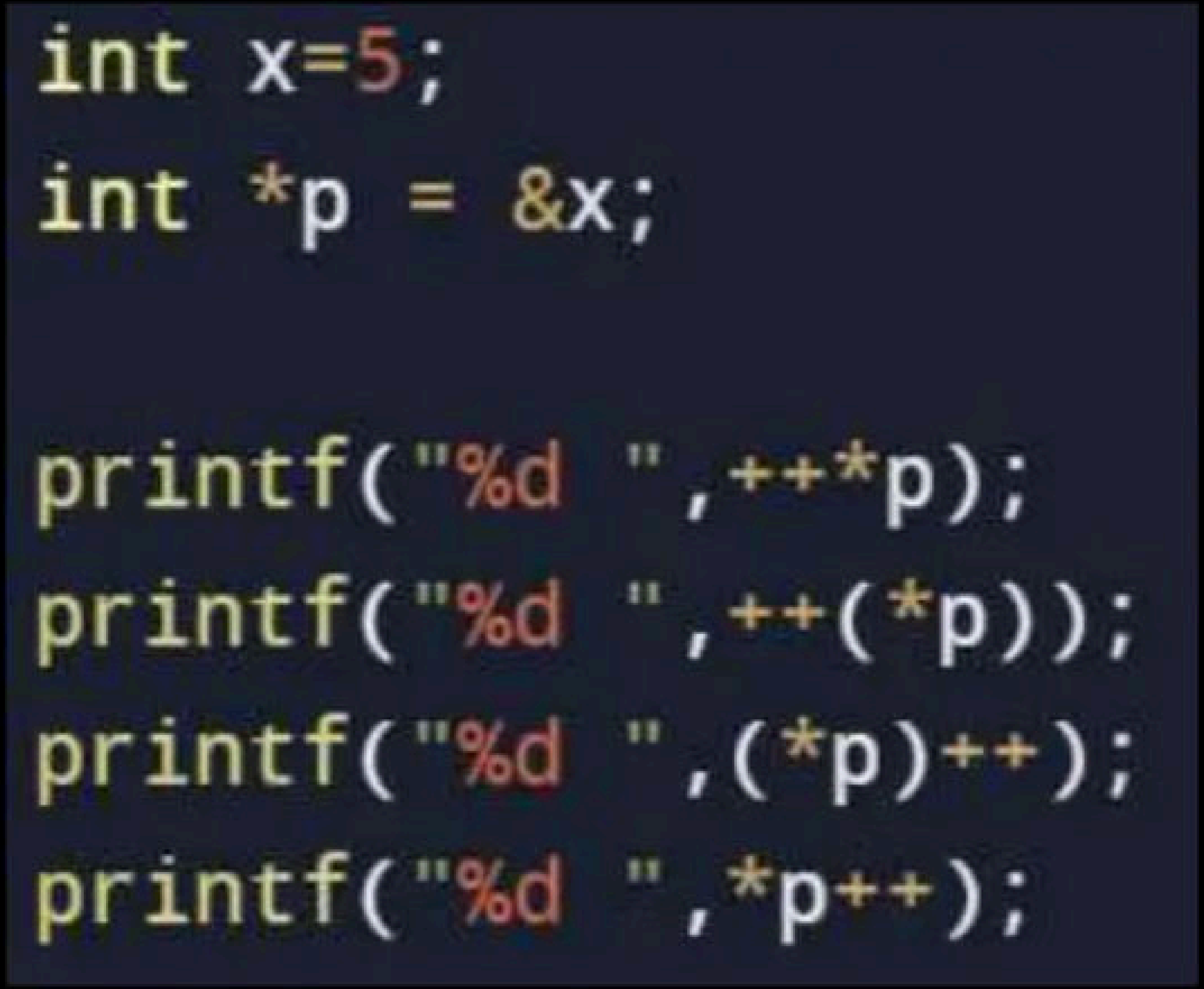
```c
#include <stdio.h>
int x;
int main() {

    return 0;
}
```

Prog2.c



```c
#include <stdio.h>
#include <Prog.c>
extern int x;
int main() {

    return 0;
}
```

Pointers

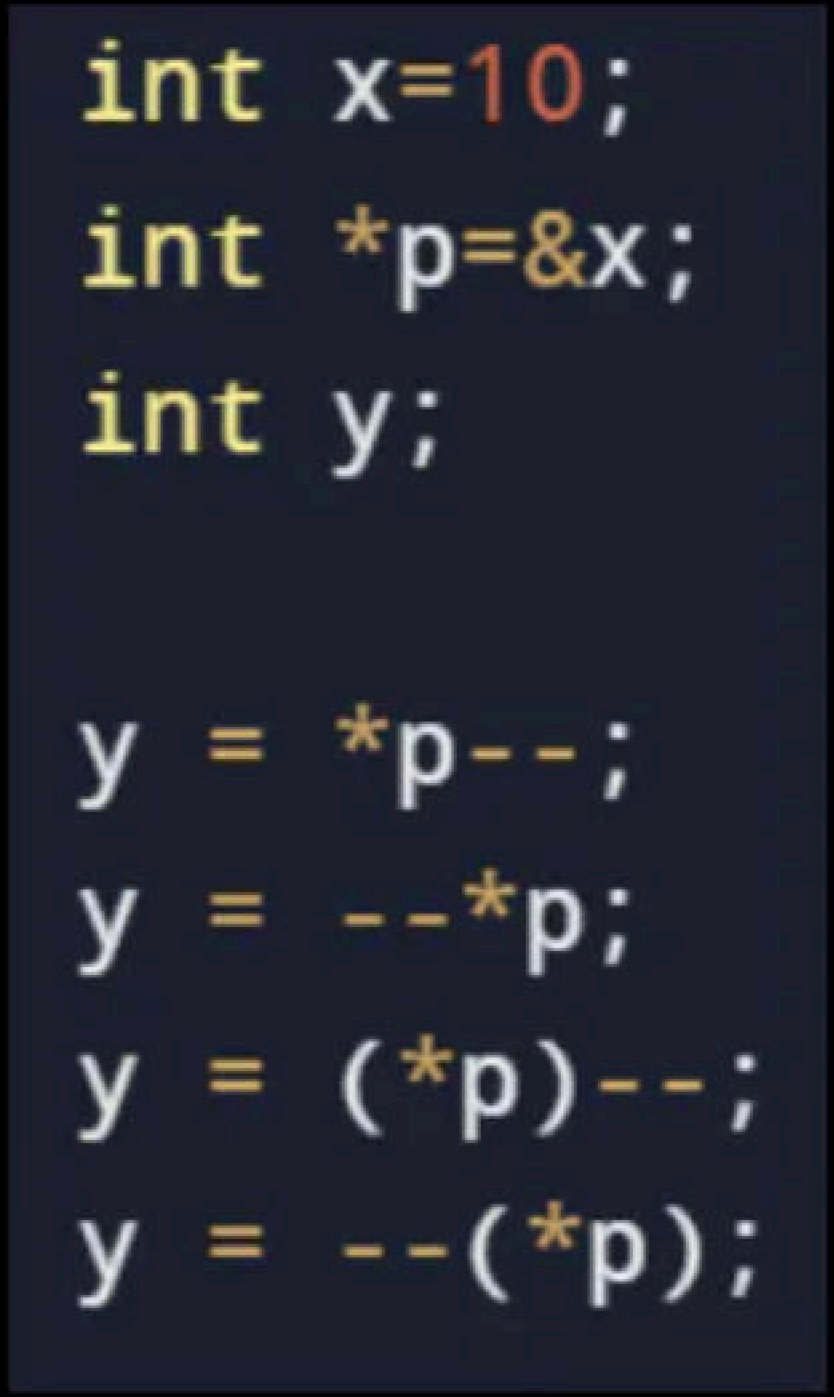


```
int x=5;
int *p = &x;


printf("%d ",++*p);
printf("%d ",++(*p));
printf("%d ",(*p)++);
printf("%d ",*p++);
```

Question

Calculate value of y, for each of the following individual case.

Assume address of x is 500 and address of p is 1100.



```
int x=10;
int *p=&x;
int y;

y = *p--;
y = --*p;
y = (*p)--;
y = --(*p);
```

NULL Pointer

String

Literals and Constants

Happy Learning.!