

Functions and Storage Classes - Part I

Comprehensive Course on C- Programming



CS & IT Engineering

C Programming

Functions & storage classes-I



Lecture Number- 13

By- Pankaj Sir



Topics

to be covered

1

Control Flow Statements



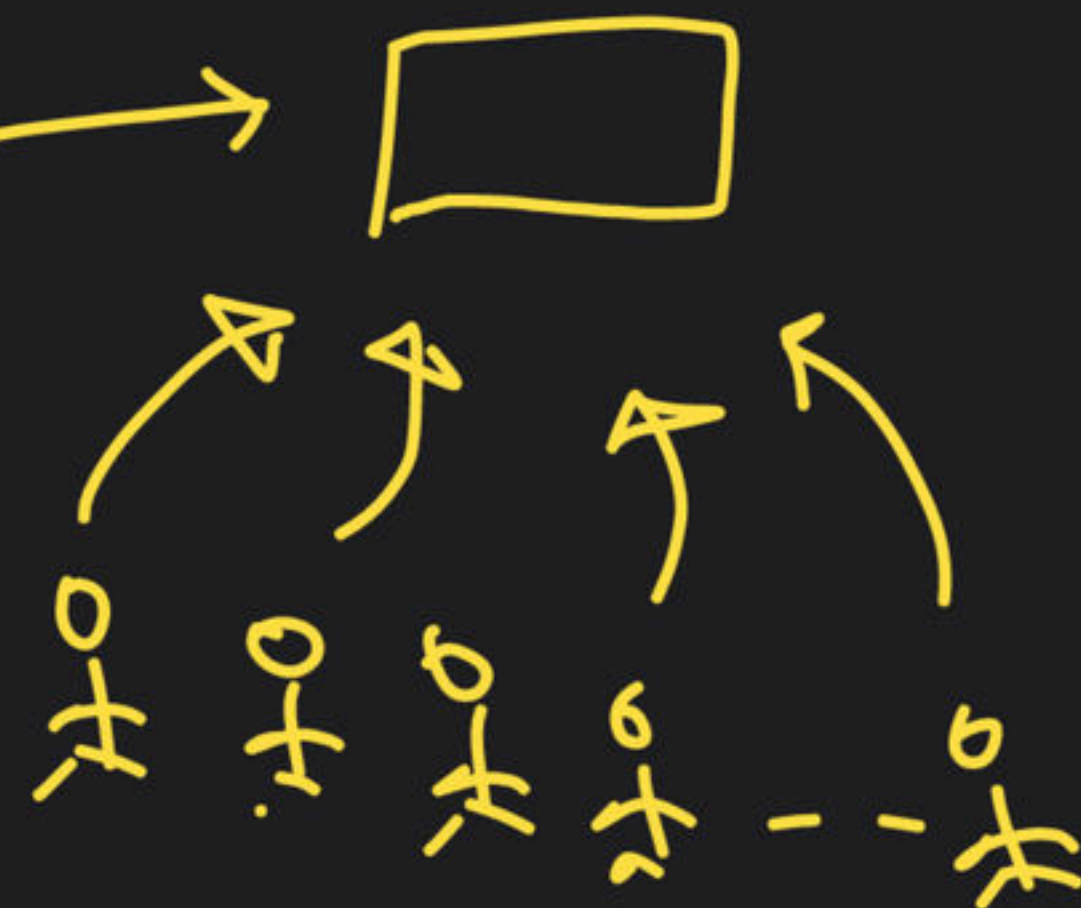
Functions

printf()
scanf()

Built-In
functions

Code

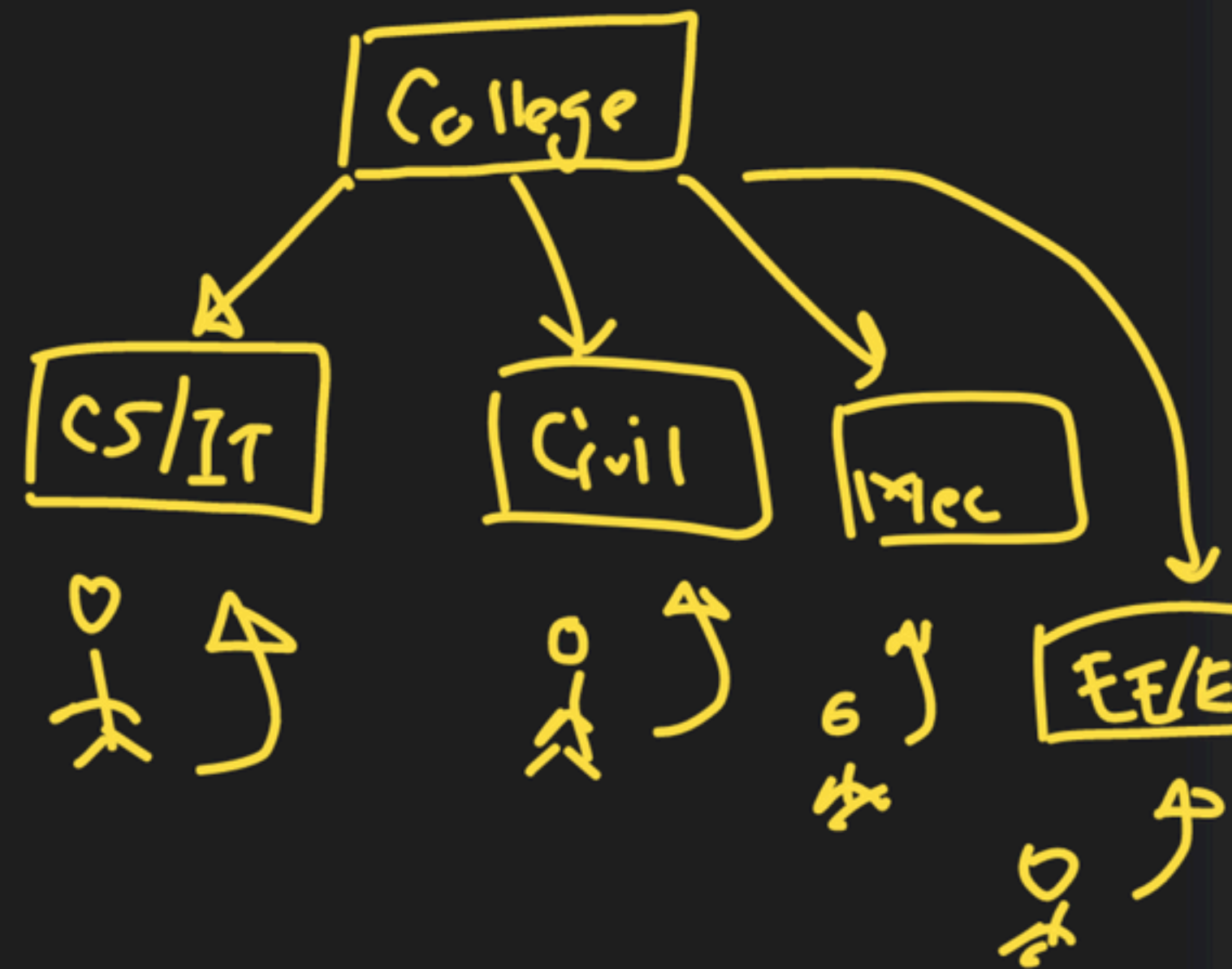
use
≈





1st way

2nd



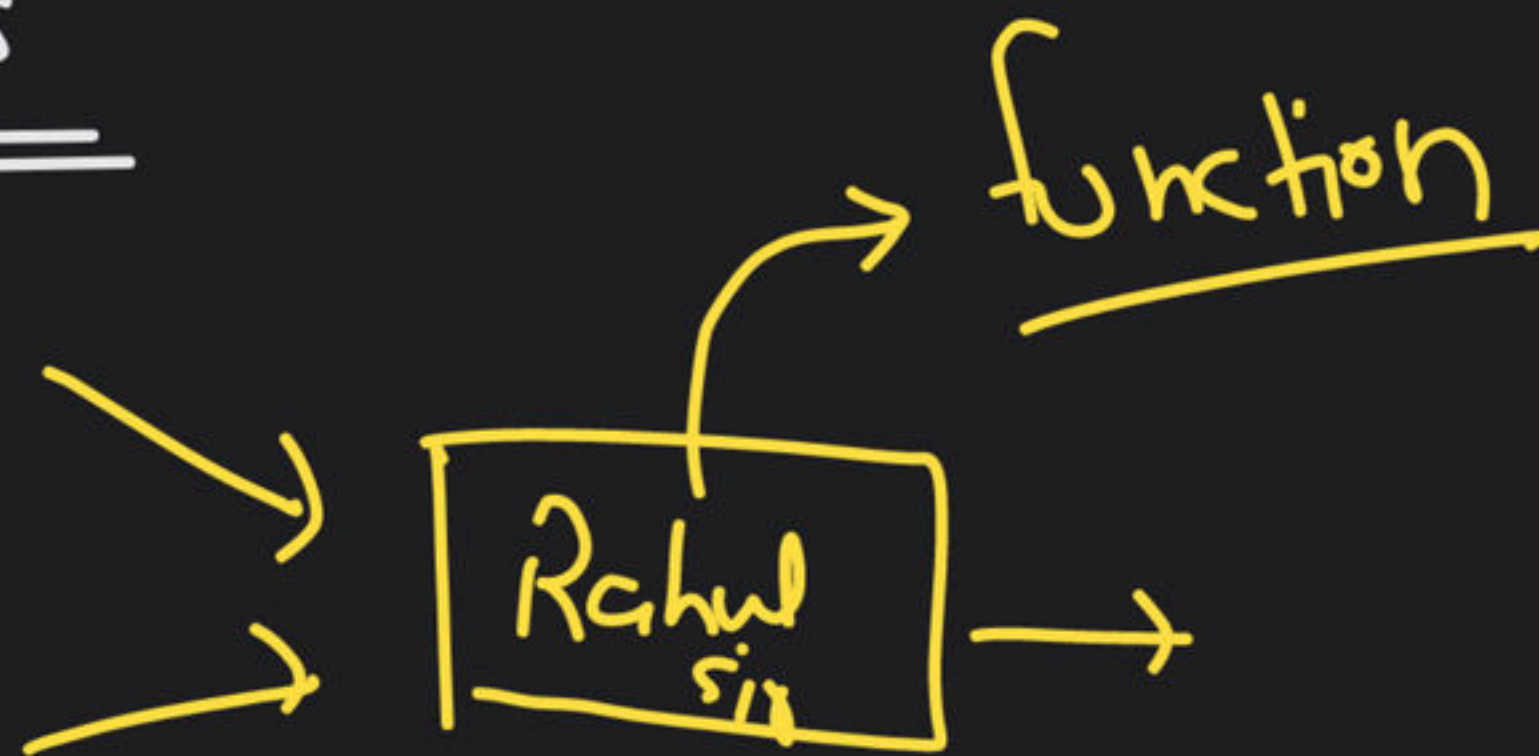
unacademy
g n complete
code
(uncorrect)

Functions

```
#include <stdio.h>
```

```
void main() {
```

```
    int a = 10, b = 20, ans;
```



gncornicode (gncorrect)

```
void main() {
```

```
int a = 10, b = 20, ans;
```

```
ans = Rahul-sir(a, b);
```

```
printf("%d", ans);
```

```
}
```

Rahul-sir(int x, int y)

{

int temp;

temp = x * y;

return temp;

x
10

y
20

200

temp

ans

200

void main() {

int a = 10, b = 20, ans;

ans = ²⁰⁰ RahulSir(¹⁰a, ²⁰b);

printf("%.d", ans); ✓

}

RahulSir(int x, int y)

x
10

y
20

200

temp

{

int temp; ✓


temp = x * y; ✓

return temp; ✓

}

```
#include <stdio.h>
void main() {
    pf("%.1d", a);
}
```

Pankaj.c

→ Ud ke last page  disha
using without declaration


```
void main() {
```

```
printf("Hello");
```

```
}
```

Pankaj.c

To avoid
C.E

forward
declaration

```
#include <stdio.h>
```

```
void main() {
```

```
printf("Hello");
```

```
}
```



```
#include <stdio.h>
```

```
void main() {
```

```
    int a=10, b=20, ans;
```

```
    ans = mul(a, b); // use // call
```

```
    printf("%d", ans);
```

```
}
```

```
int mul(int x, int y)
{
    int temp;
    temp = x * y;
    return temp;
}
```

↑
define
↓

```
#include <stdio.h>
```

```
int mul(int, int);
```

```
void main{
```

```
int a=10, b=20, ans;
```

```
ans = mul(a, b); // use // call
```

```
printf("%.1d", ans);
```

```
}
```

forward declaration

function header

```
int mul(int x, int y)
```

define

```
{  
int temp;  
temp = x * y;  
return temp;  
}
```



```
#include <stdio.h>
int mul(int p, int q);
void main() {
    int a=10, b=20, ans;
    ans = mul(a, b); // use // call
    printf("%d", ans);
}
```

forward dec. for info. to avoid C.E

No
memory
is
allocated

int mul(int x, int y)

```
{
    int temp;
    temp = x * y;
    return temp;
}
```

define


```
#include <stdio.h>
```

```
int mul (int x, int y)
```

```
{  
    int temp;  
    temp = x * y;  
    return temp;  
}
```

define

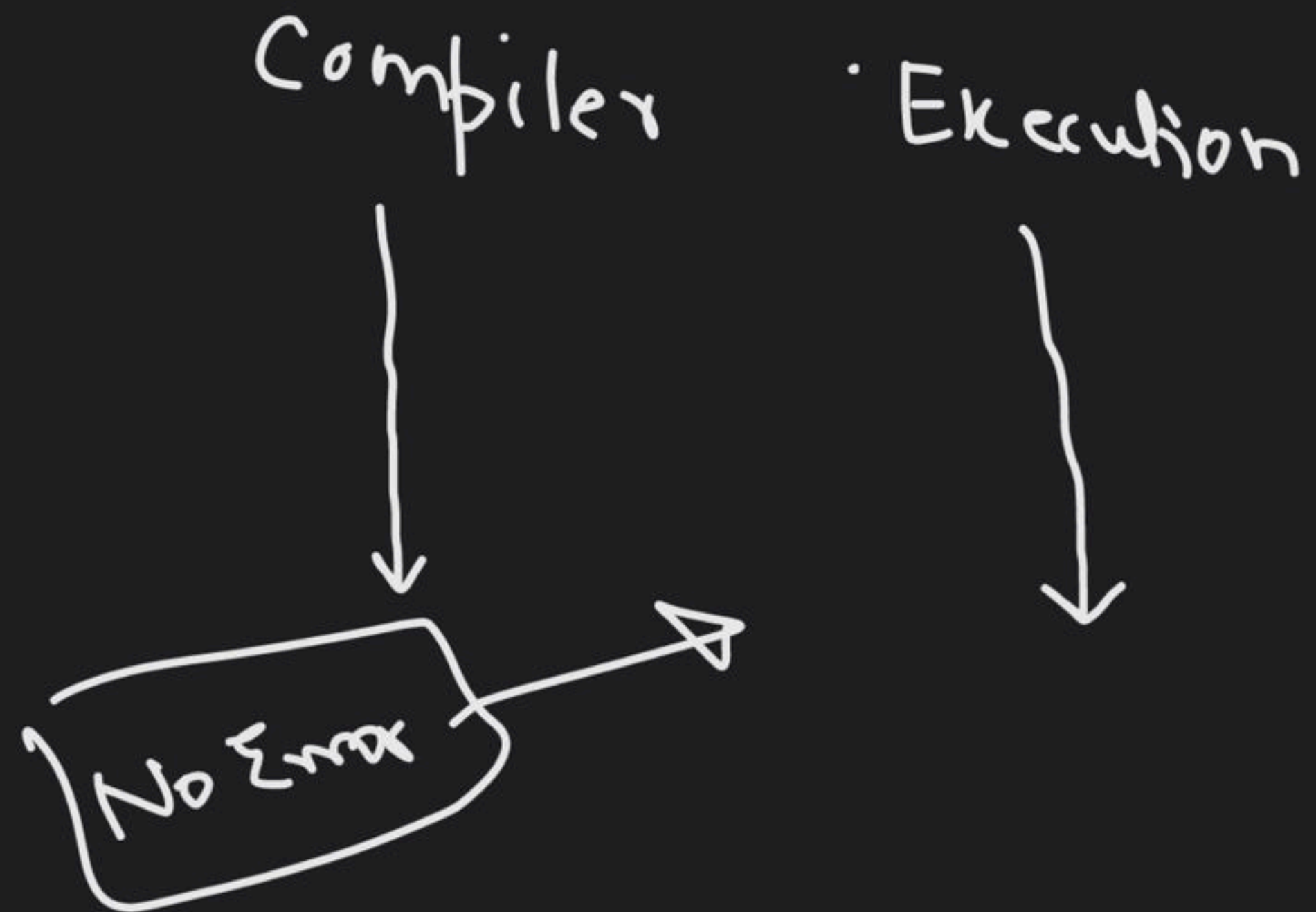
```
void main() {
```


```
    int a=10, b=20, ans;
```

```
    ans = mul(a, b);
```

```
    printf("%d", ans);
```

```
}
```



 int x ;

short x ; ✓

by default
it is signed int

by - default
⇒ short int

by default \Rightarrow int

```
#include <stdio.h>
```

```
mul(int, int);
```

```
void main() {
```

```
int a = 10, b = 20, ans;
```

```
ans = mul(a, b);
```

```
printf("%d", ans);
```

```
}
```

```
int mul(int x, int y)  
{
```

```
int temp;
```

```
temp = x * y;
```

```
return temp;
```

```
}
```

```
#include <stdio.h>
```

```
void main() {
```

```
    int a = 10, b = 20, ans;
```

```
    ans = mul(a, b);
```

```
    pf("%d", ans);
```

```
}
```

int

```
mul(int x, int y)
```

```
{
```

```
    int temp;
```

```
    temp = x * y;
```

```
    return temp;
```

```
}
```

The return
type of
mul() is
int



Happy


```
#include <stdio.h>
```

```
void main() {
```

```
    int a = 3;
```

```
    double b;
```

```
    b = f(a);
```

```
    printf("%d", b);
```

```
}
```

The return
type of func.
f() is int

vd Re laak }

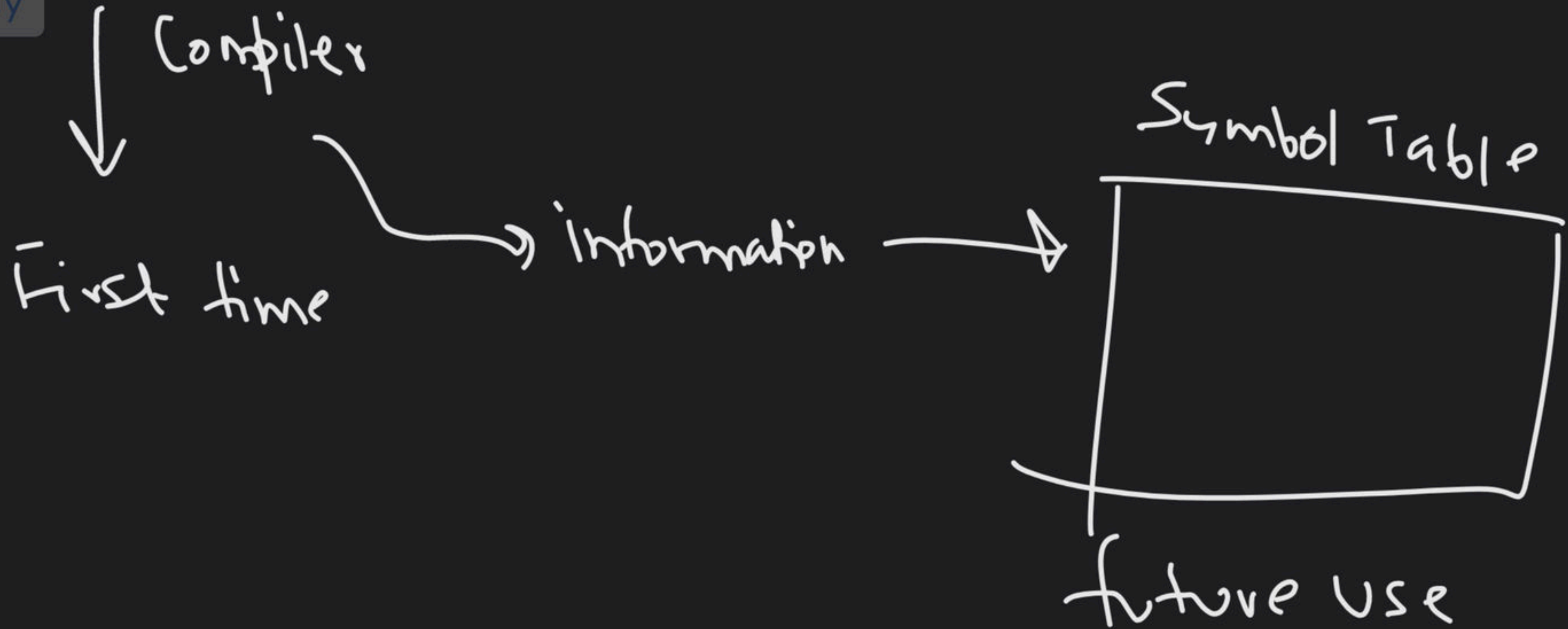
```
double f(int x)
```

```
{
```

```
    double y = 2.38;
```

```
    return x * y;
```

```
}
```

```
#include <stdio.h>
```

int

```
void main{
```

```
    ✓  
    ✓  
    ans = f(10, 20, 30);  
    ✓  
}
```



~~int a;~~
~~a = 7.2;~~

→ forward declaration

#include <stdio.h>

void main() {

printf("Hello"); // call

}

} Pf → code/define

② Macro Expansion
① file inclusion

```
#include <stdio.h>
#define MAX 20
void main() {
    printf("%d", MAX + MAX);
}
```

P.C

Pre
processing

```
int printf(const char*,...);
...
void main() {
    pf("%d", 20 + 20);
}
```

P.i

compile

Assembly
language

P.S

Assembly

M/C
inst.

P.O

printf.o



scanf.o

Linker

only
knows
main

a.out

Executable
64
runnable

(Permanent Code) loader

Q1. Can we compile a prog, without main()?

Yes.

Q2. Can we execute a prog without main?

No

↳ Linker

#inc _____

```
void main() {
```

```
    printf("Hello");
```

```
}
```

Are we using the value returned by printf?

NO

#in _____

```
void main() {
```

```
    int i;
```

```
    i = printf("Hello");
```

```
    pf("%.d", i);
```

```
}
```

we are using the value returned by printf?


```
int mul(int x, int y);
```

```
void main() {
```

```
    int a = 10, b = 20;
```

```
    mul(10a, 20b);
```

```
}
```

```
void main() {
```

```
    200;
```

→ No Error

```
int mul(int x, int y)
```

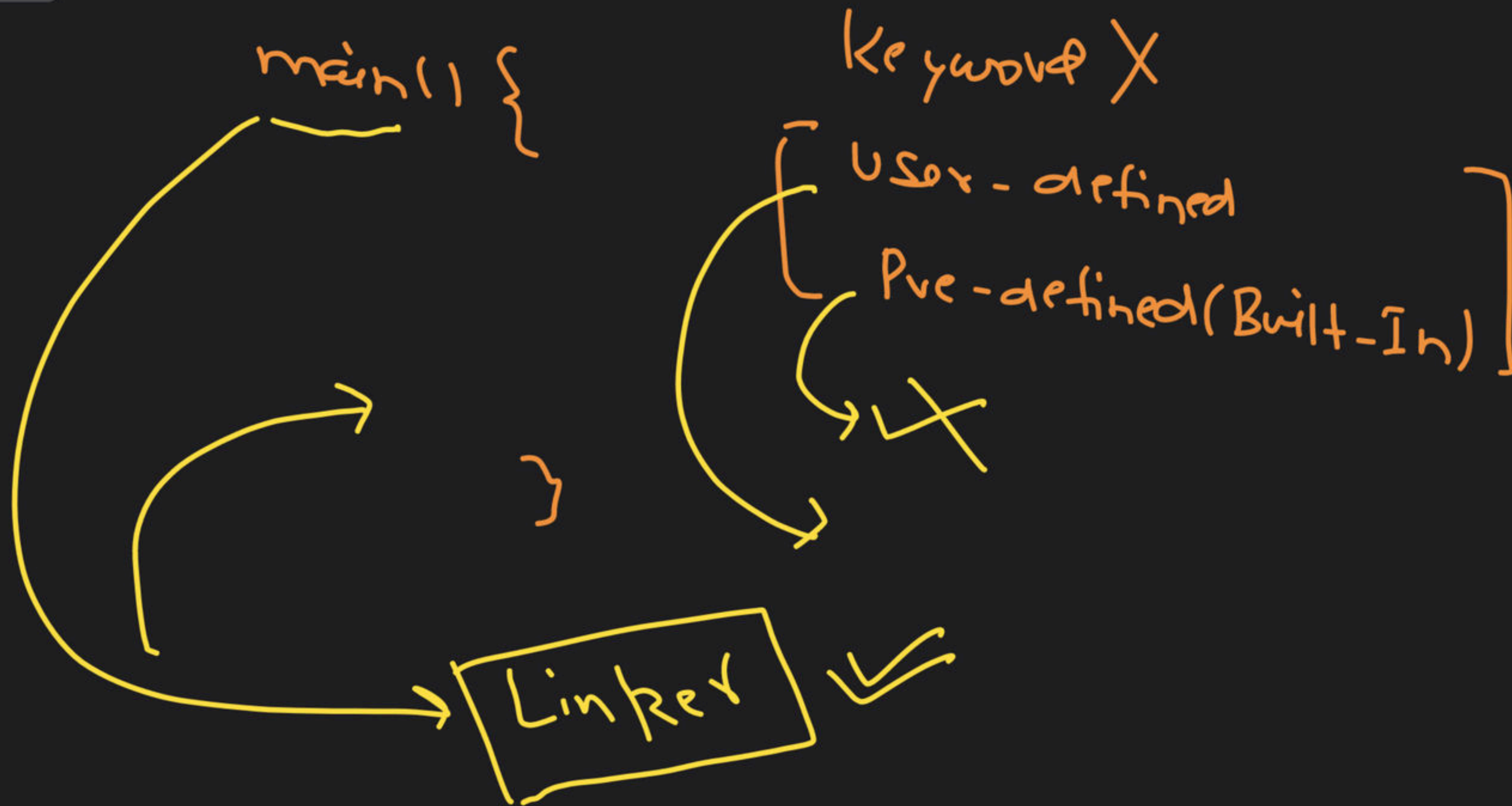
```
{
```

```
    int temp;
```

```
    temp = x * y;
```

```
    return temp;
```

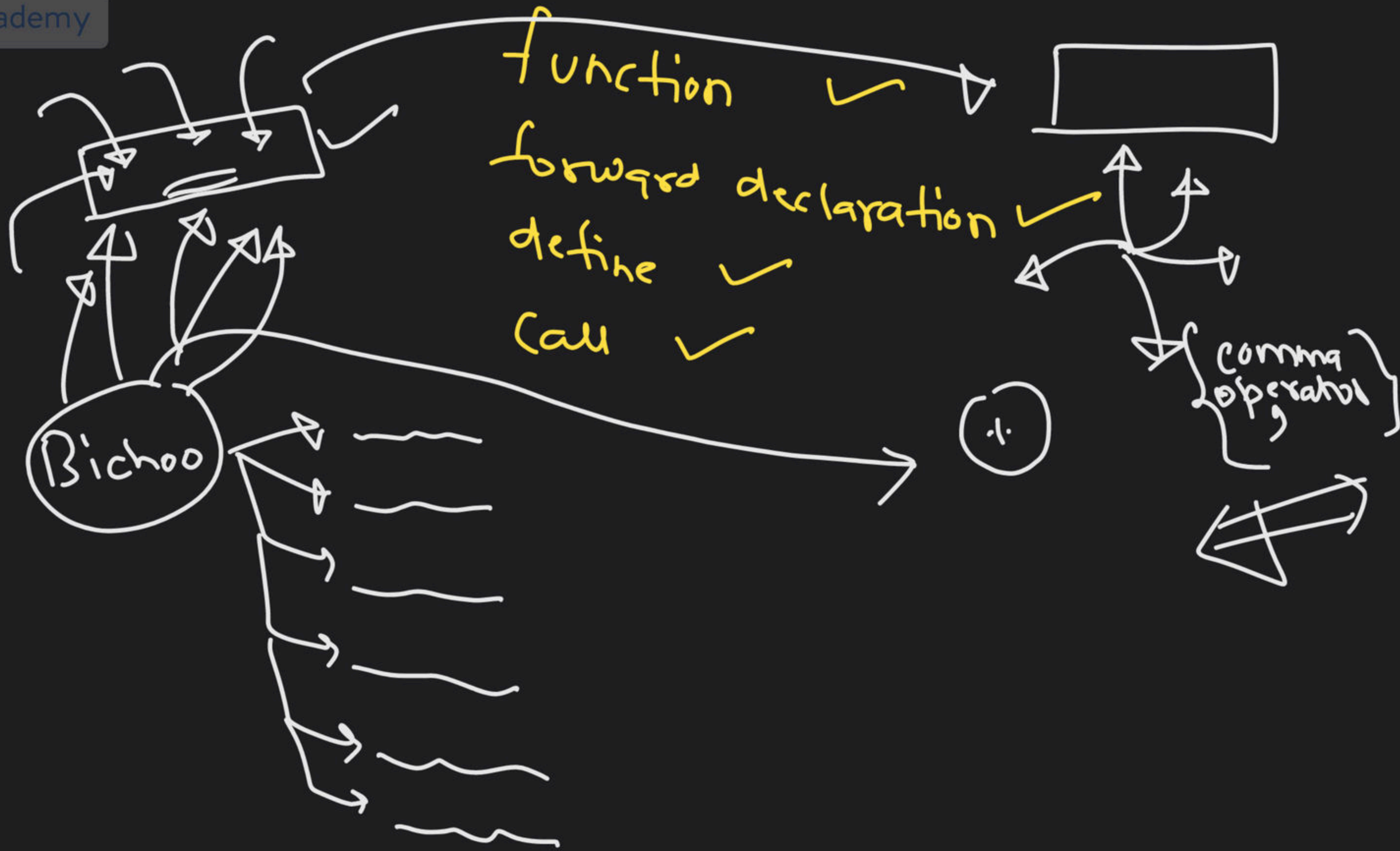
```
}
```

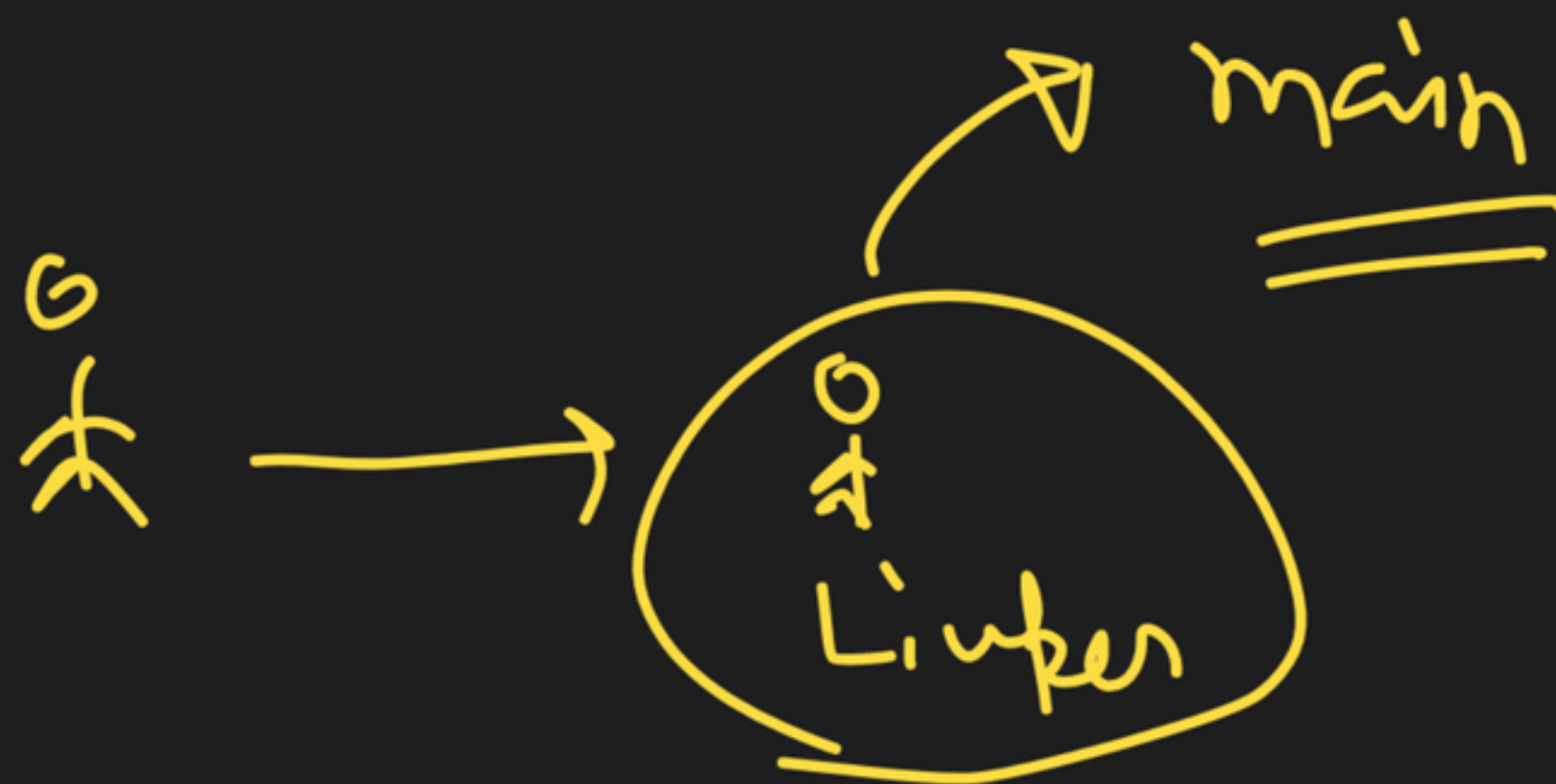


void ^{just} main() {

Add 1

— Add 1
 {
 }





Programming

```
void main() {  
    int a;  
    200;  
    a = 200;  
}
```

The image shows a handwritten C program snippet. The first line is `void main() {`. The second line is `int a;`. The third line is `200;`, which is circled in green. The fourth line is `a = 200;`, where the variable `a` is circled in white. A white arrow points from the circled `a` to the circled `200` in the line above. The fifth line is `}`.





THANK YOU!

Here's to a cracking journey ahead!