

CS & IT ENGINEERING

Programming in C

Strings


Lec- 02



By- Pankaj Sharma sir



TOPICS TO BE
COVERED



Strings-2

```
char arr1[] = "Panpaj";
```

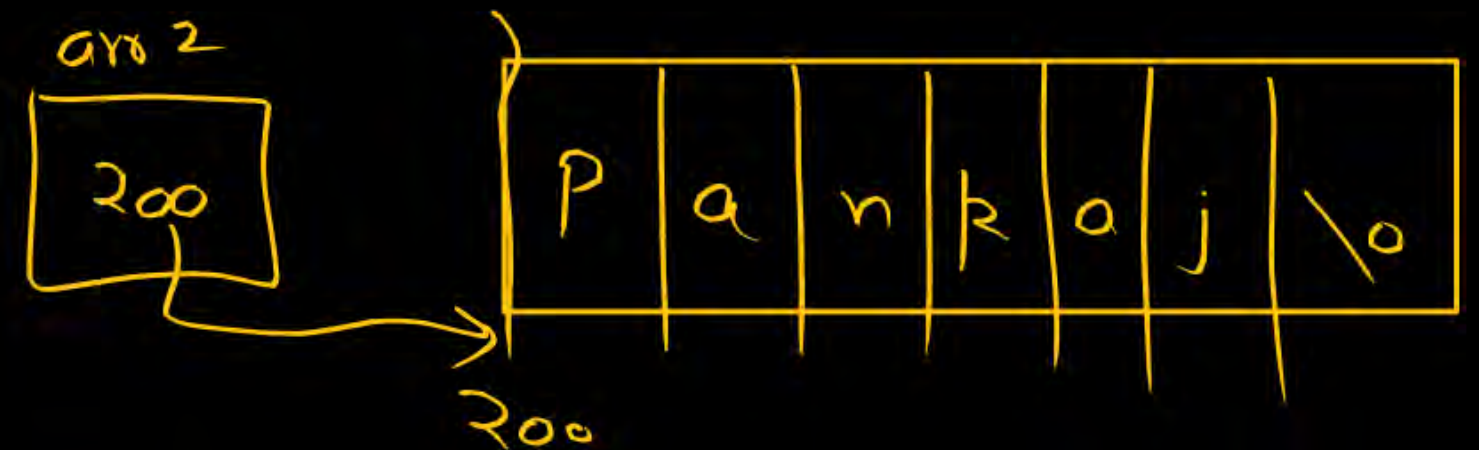
```
char arr2[] = "Panpaj";
```

```
if (arr1 == arr2)
```

```
    pf("Yes");
```

```
else
```

```
    pf("No");
```



```
char arr1[] = "Pankaj";
```

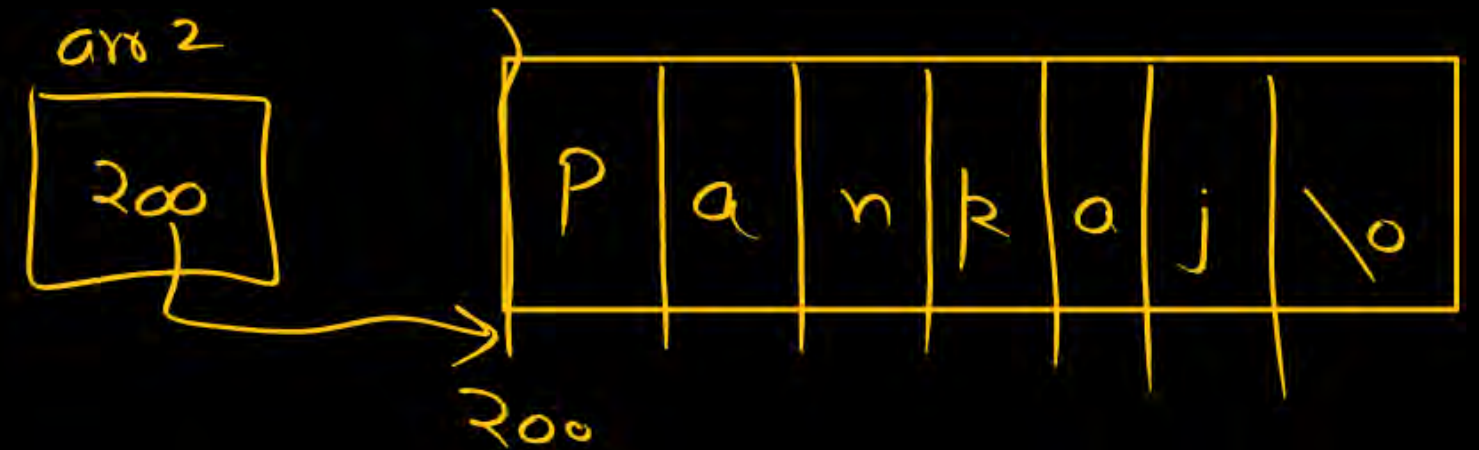
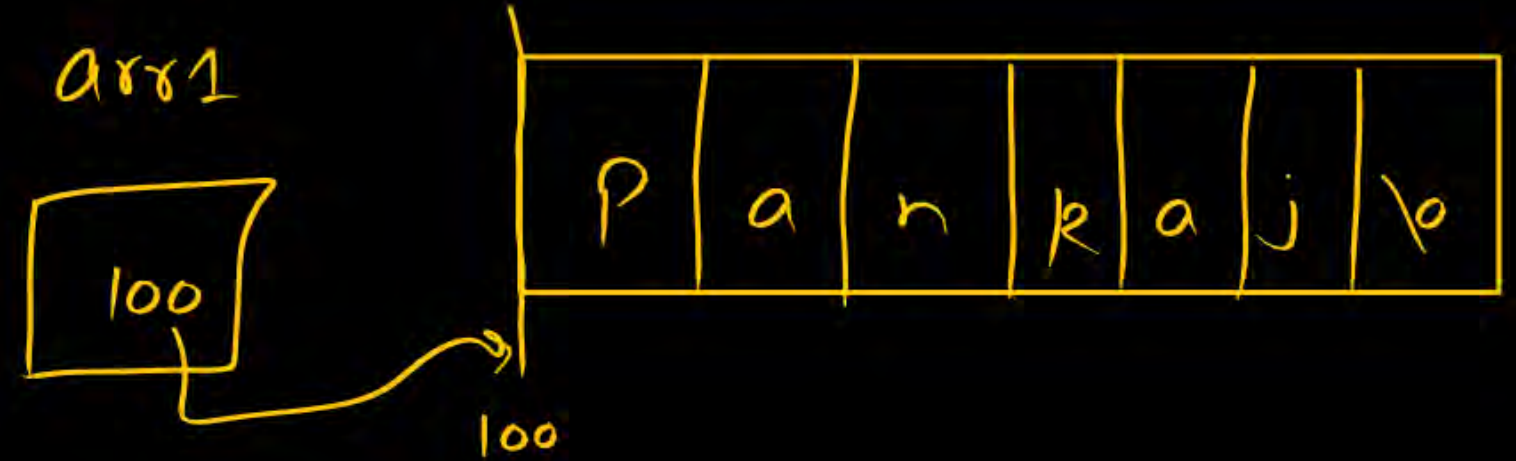
```
char arr2[] = "Pankaj";
```

```
if (*arr1 == *arr2)
```

```
    pf("Yes"); ✓
```

```
else
```

```
    pf("No");
```




```
char *p = "Pankaj";
```

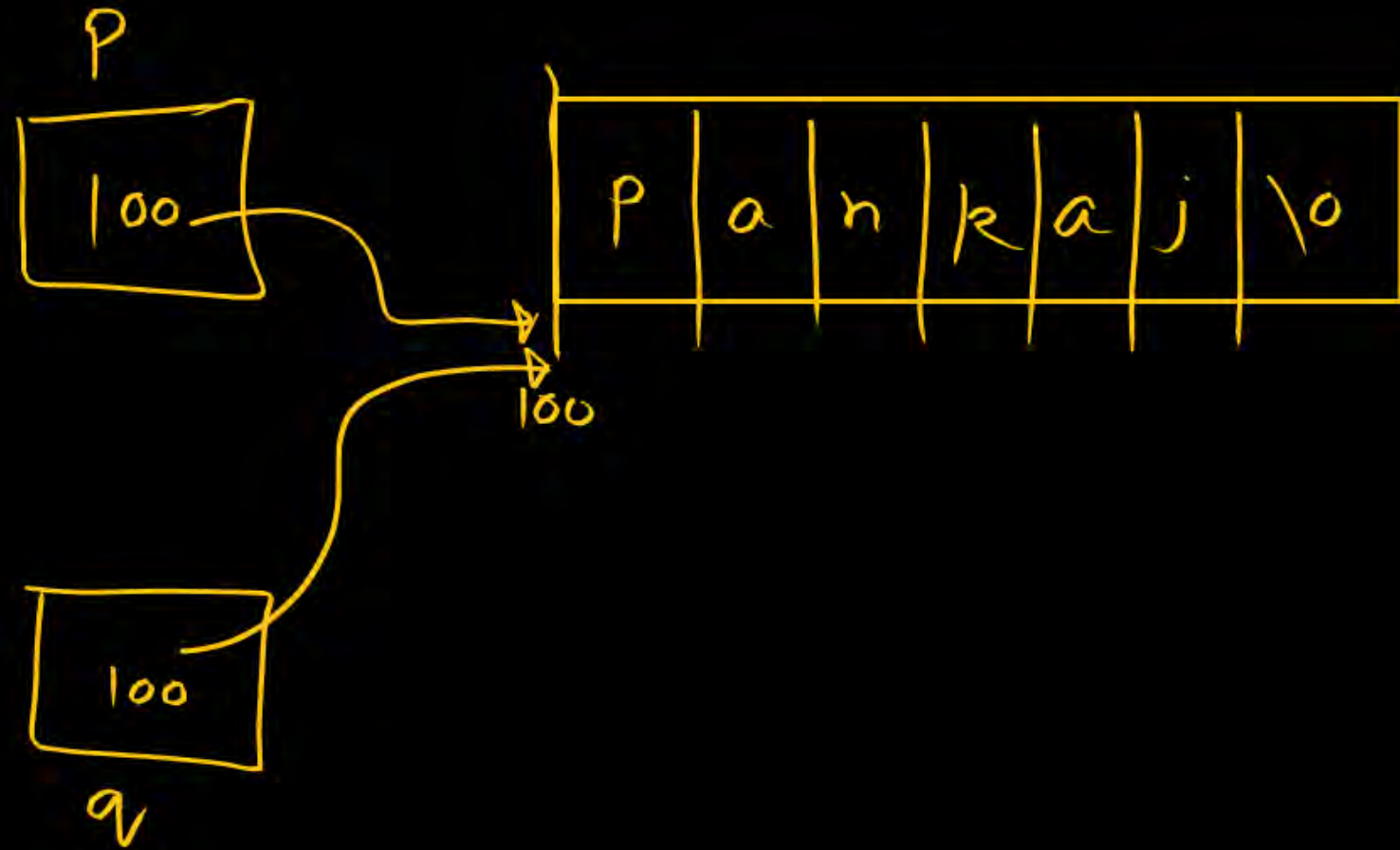
```
char *q = "Pankaj"; ✓
```

```
if (p == q)
```

```
    printf("Yes");
```

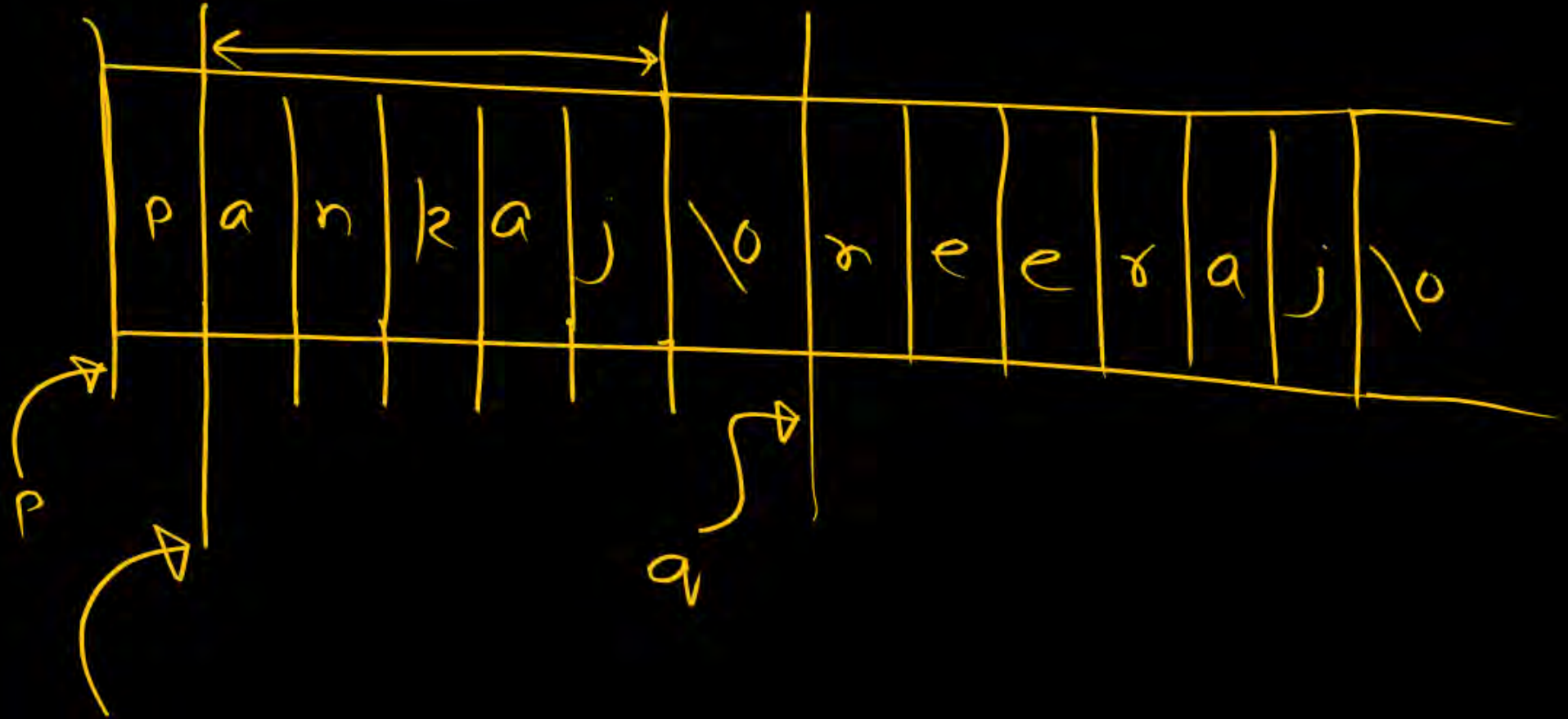
```
else
```

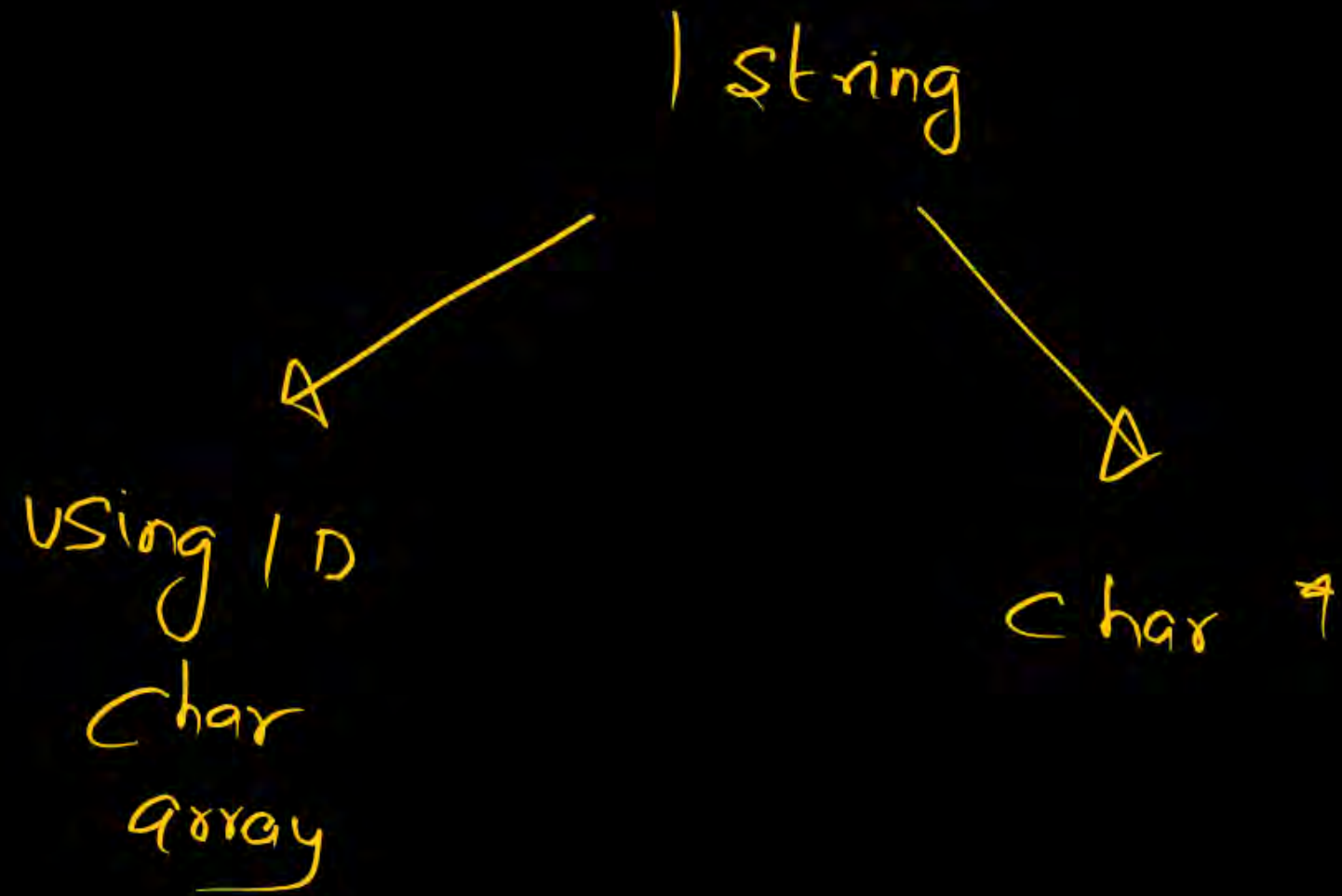
```
    printf("No");
```



char *p = "ponkaj"

an/2a]





3 string

Multiple string

- ① char arr1[] = "Amit";
- ② char arr2[] = "Rahul";
- ③ char arr3[] = "Balamukunda";

char arr[3][12]

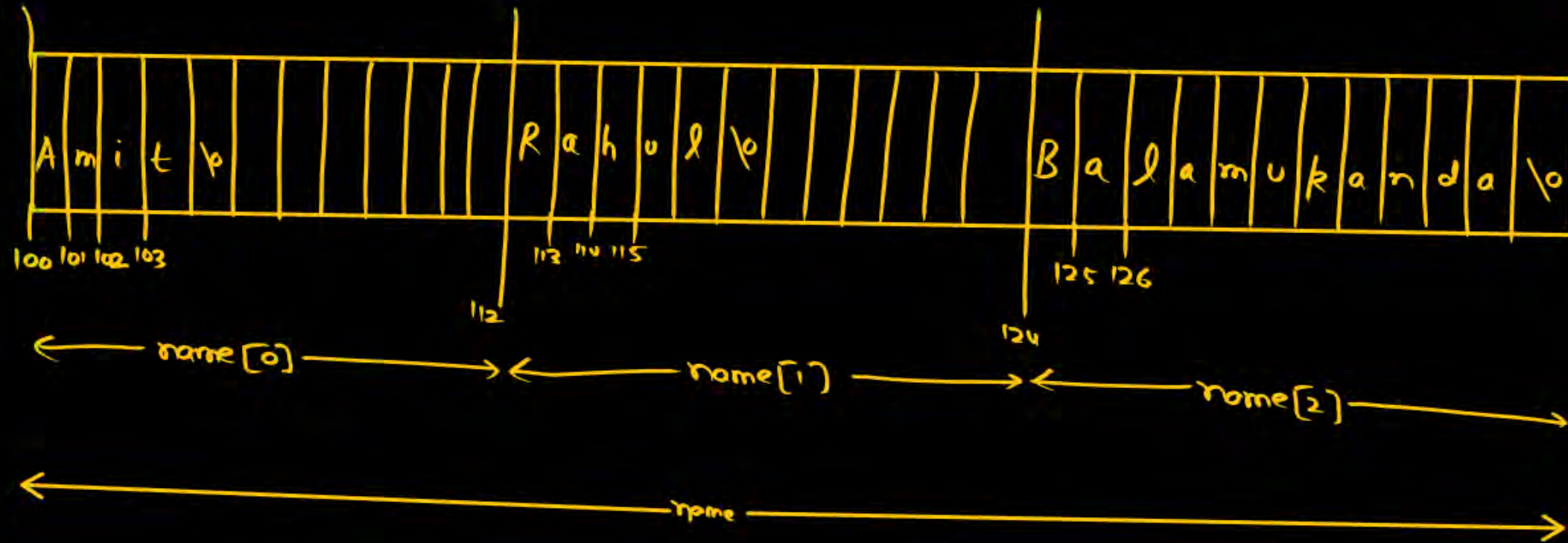
= {
 "Amit", "Rahul",
 "Balamukunda"
};

char name[3][12] = {"Amit", "Rahul", "Balamukunda"};

Add of

name[1] \Rightarrow 'R' in Rahul

name[1] \Rightarrow &name[1][0]



name[1] + 2
add. of
= 'h' in Rahul

* (name[1] + 2)
 \Rightarrow 'h' in Rahul

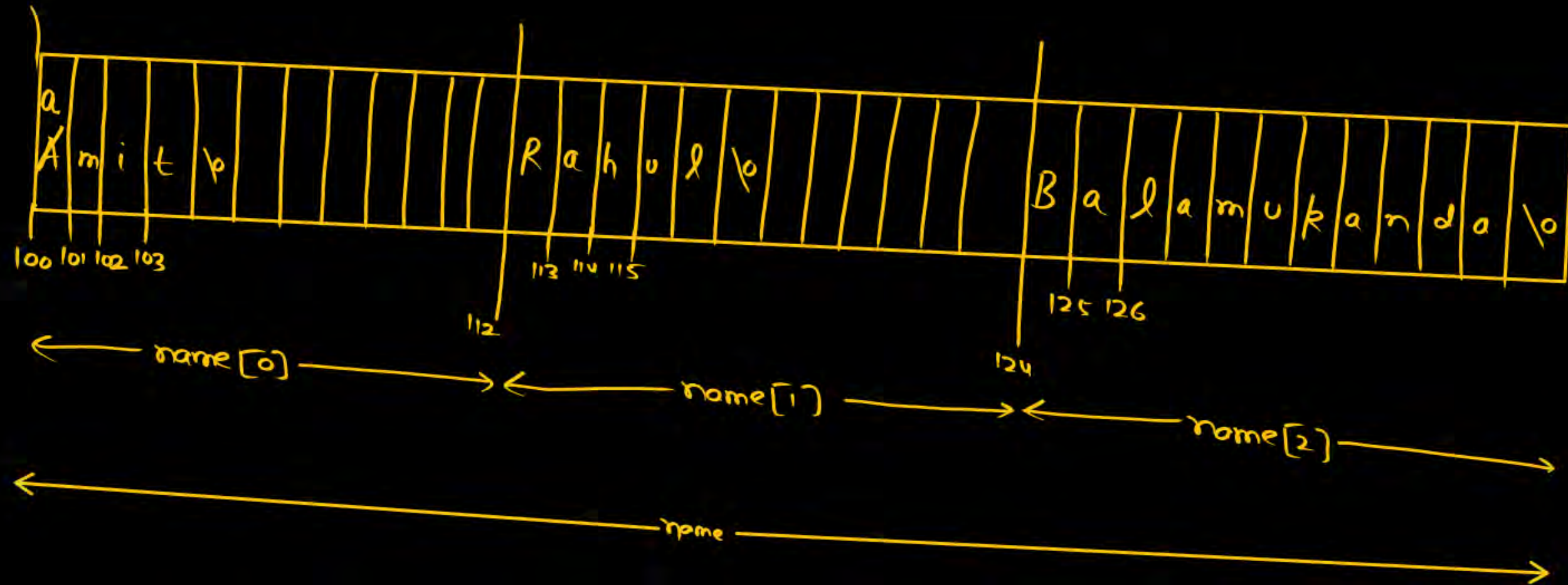
name[1][2] \Rightarrow 'h'

name[0] \Rightarrow &name[0][0] \Rightarrow Add. of 'A' in Amit

printf("%s", name[0]); Amit

printf("%s", name[0] + 2); it

`char name[3][12] = {"Amit", "Rahul", "Balamukanda"};`



`name[0] = "Pankaj";`



`name[0][0] = 'a';` ✓

char name[3][12] = { "Amit",
"Rahul",
"Balamukunda"
};

→ 4+1
→ 5+1
→ 11+1

$$\Rightarrow 5 + 6 + 12 = 23 \text{ byte}$$

→ 36 byte

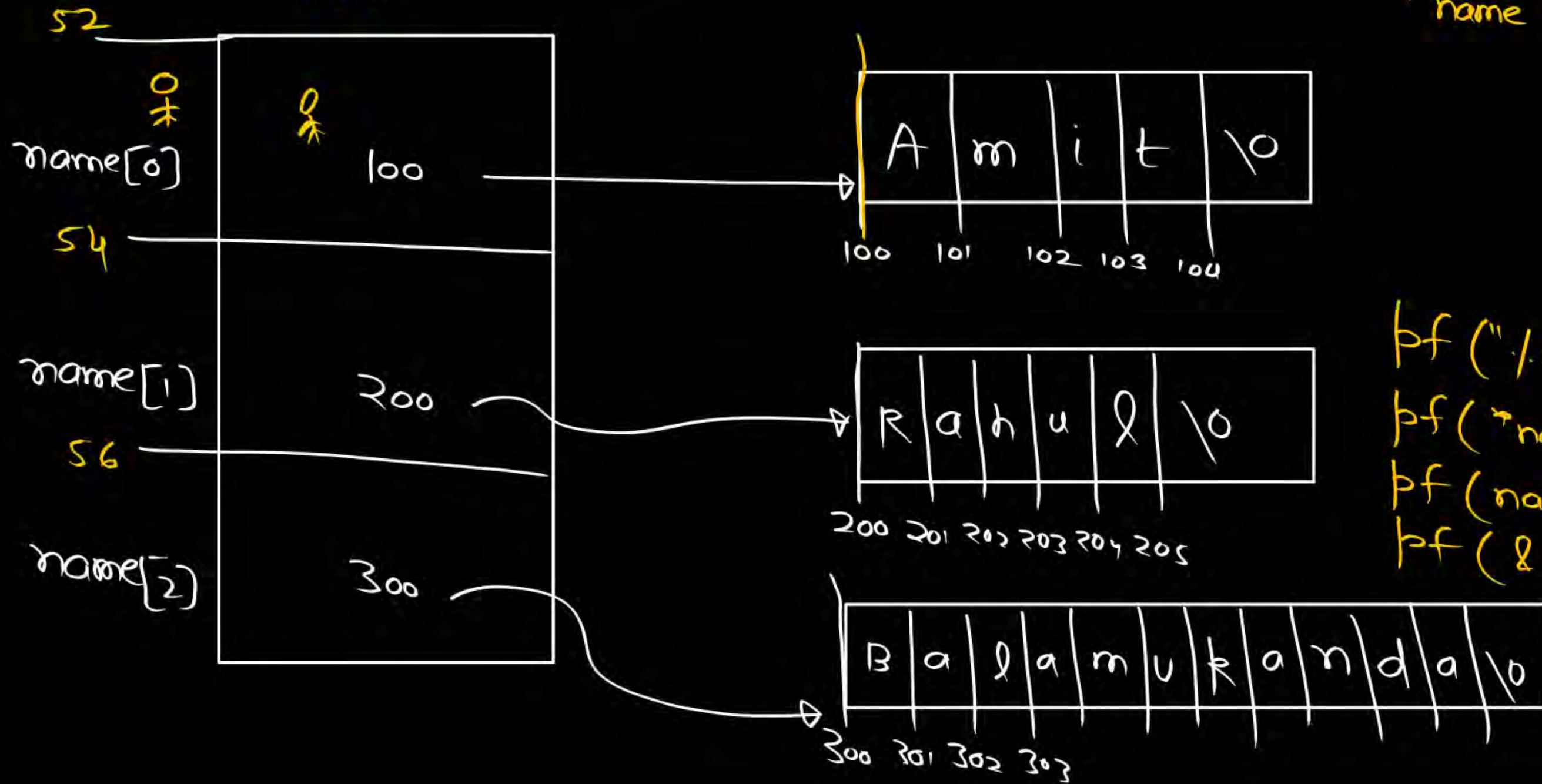
char *p = "Amit";
char *q = "Rahul";
char *s = "Balamukunda";

3 (char *)
↓

```
char *name[3]  
= {"Amit",  
   "Rahul",  
   "Balamukunda"};
```

char *name[3] = {"Amit", "Rahul", "Balamukunda"};

name



(52)
name \Rightarrow &name[0]

*name \Rightarrow */&name[0]

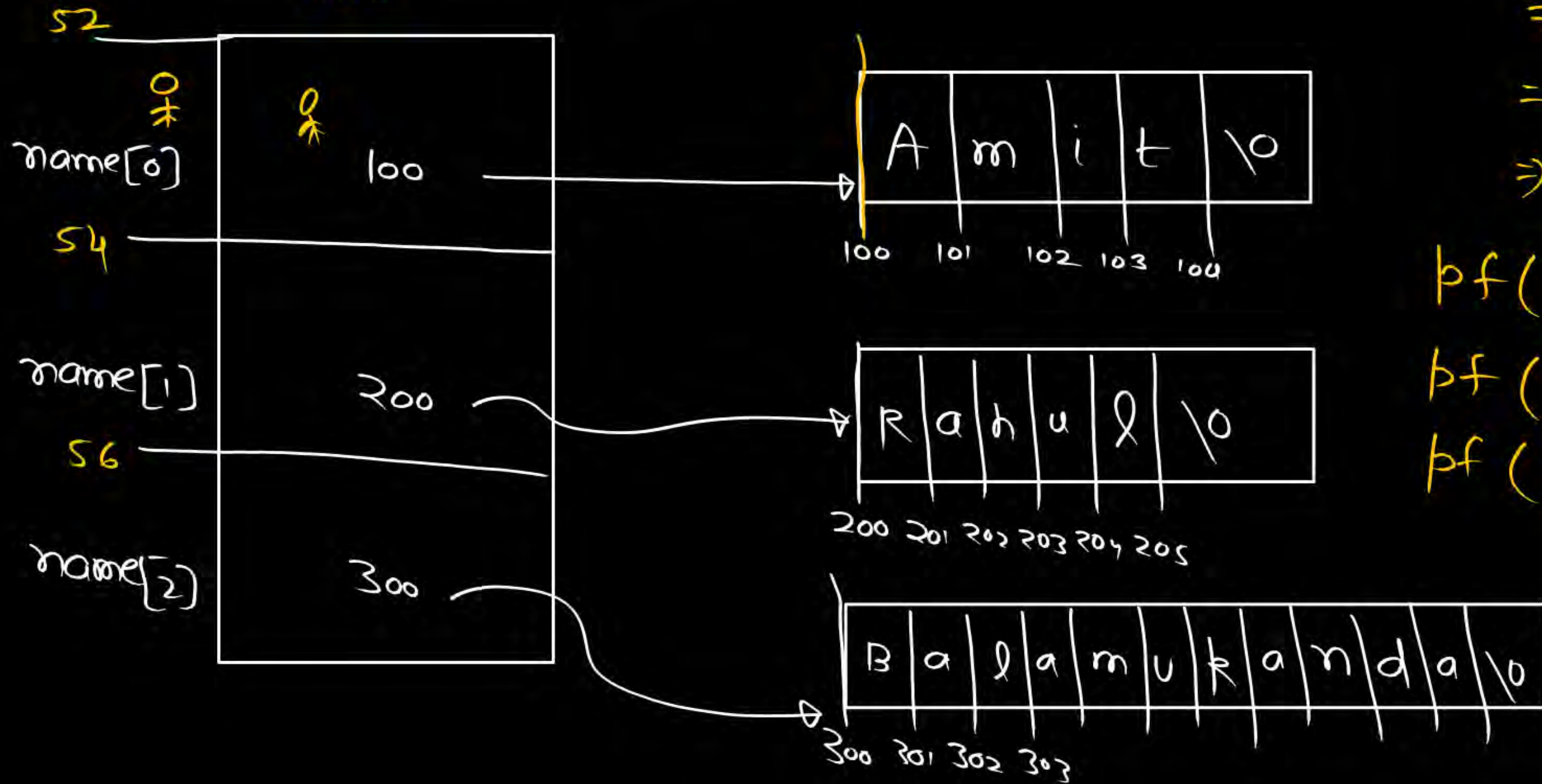
\Rightarrow name[0]
= &name[0][0]
Add of 'A'
in "Amit"

`printf("%s", *name);`
`printf(*name);`
`printf(name[0]);`
`printf(&name[0][0]);`

Amit

char *name[3] = {"Amit", "Rahul", "Balamukunda"};

name



*name + 1

\Rightarrow `name[0] + 1`

$=$ `&name[0][0] + 1`

\Rightarrow 'm' address

`pf(*name + 1);`

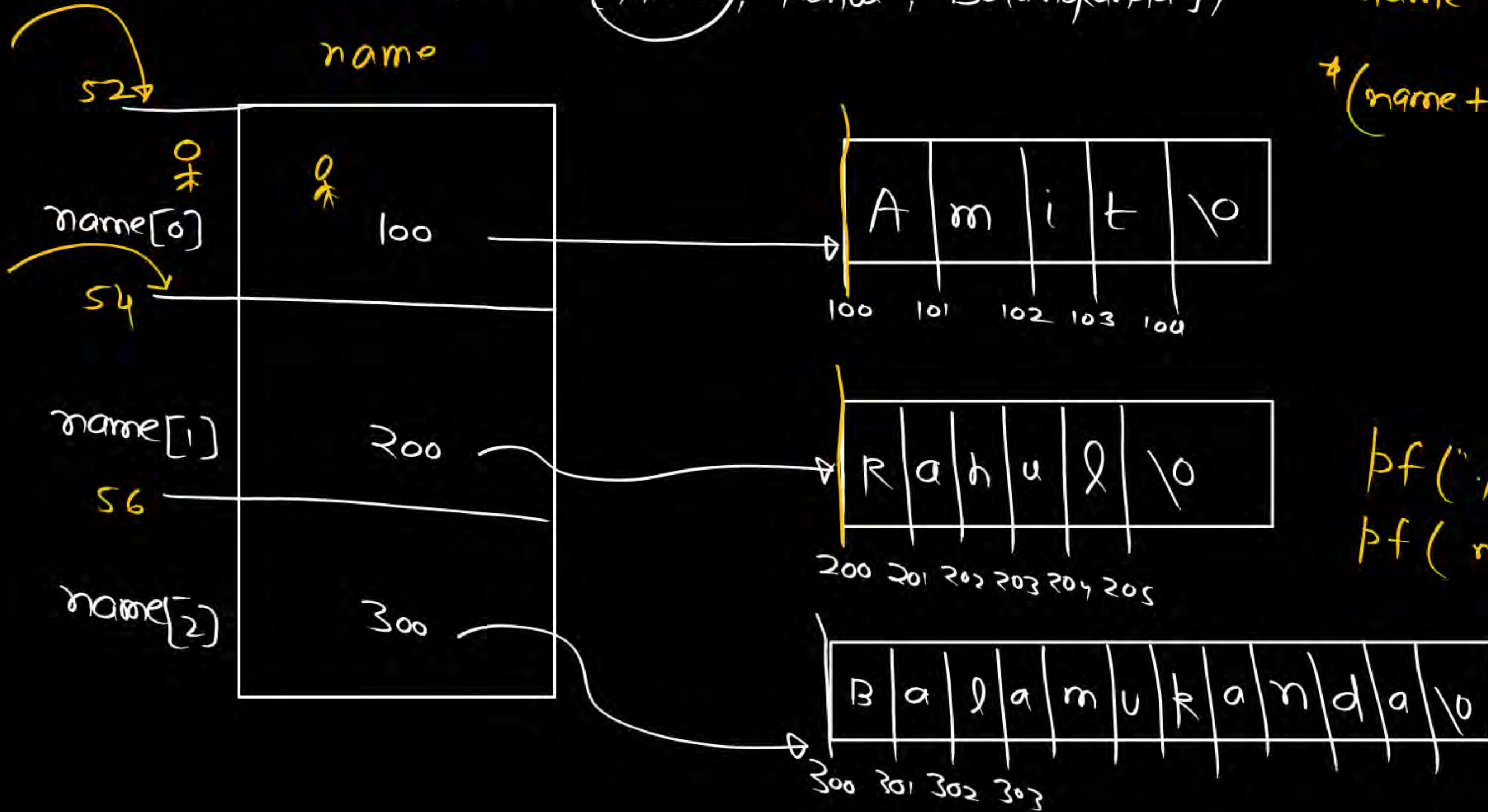
`pf(&name[0][1]);`

`pf(name[0] + 1);`

mit

char *name[3] = {"Amit", "Rahul", "Balamukunda"};

name



name + 1 \Rightarrow &name[1]

* (name + 1) = ~~*name[1]~~

= name[1]

\Rightarrow add of 'R' in "Rahul"

pf("./s", name[1]);

pf(name[1]);

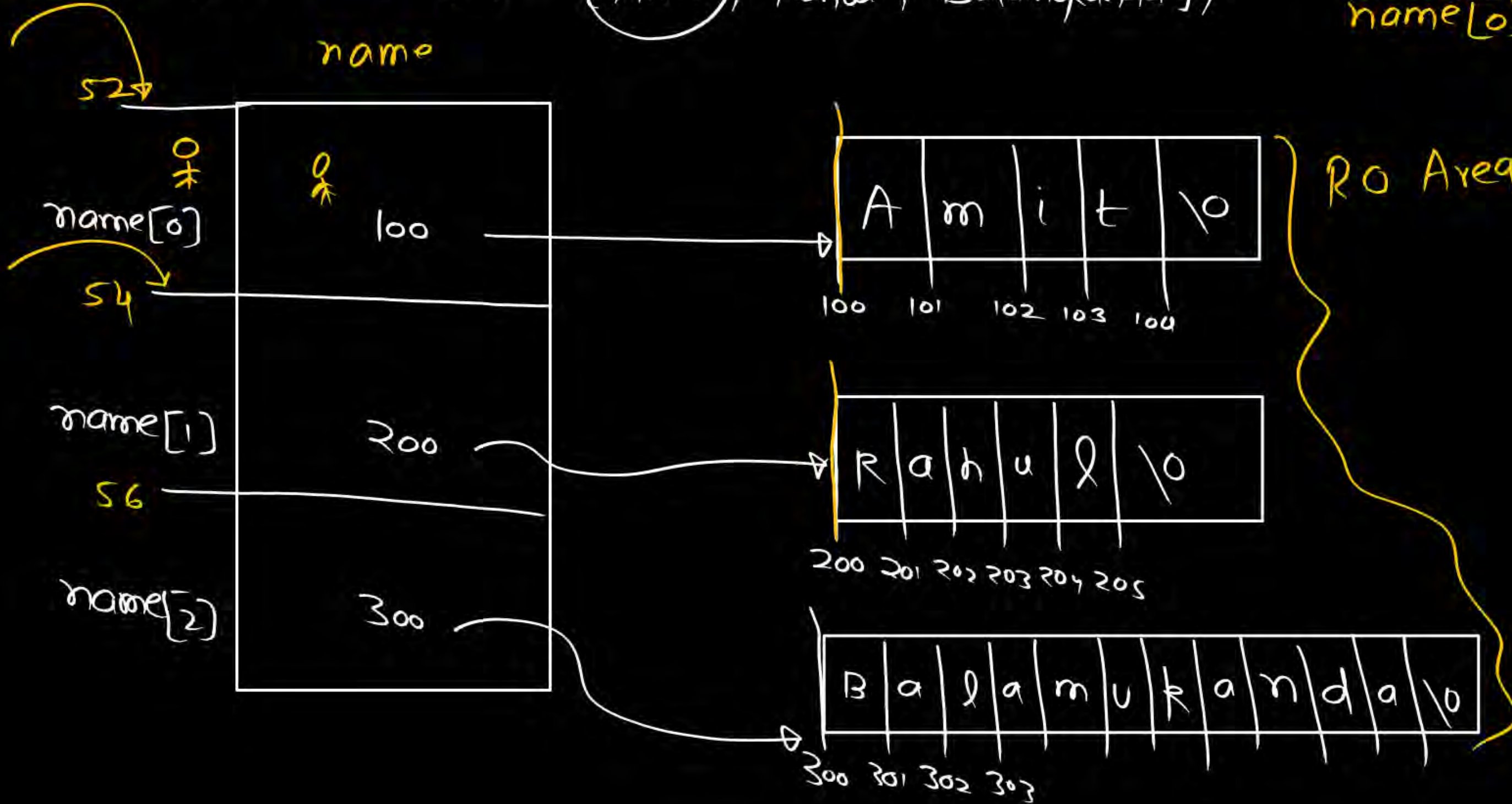
pf(* (name + 1));

Rahul

char *name[3] = {"Amit", "Rahul", "Balamukunda"};

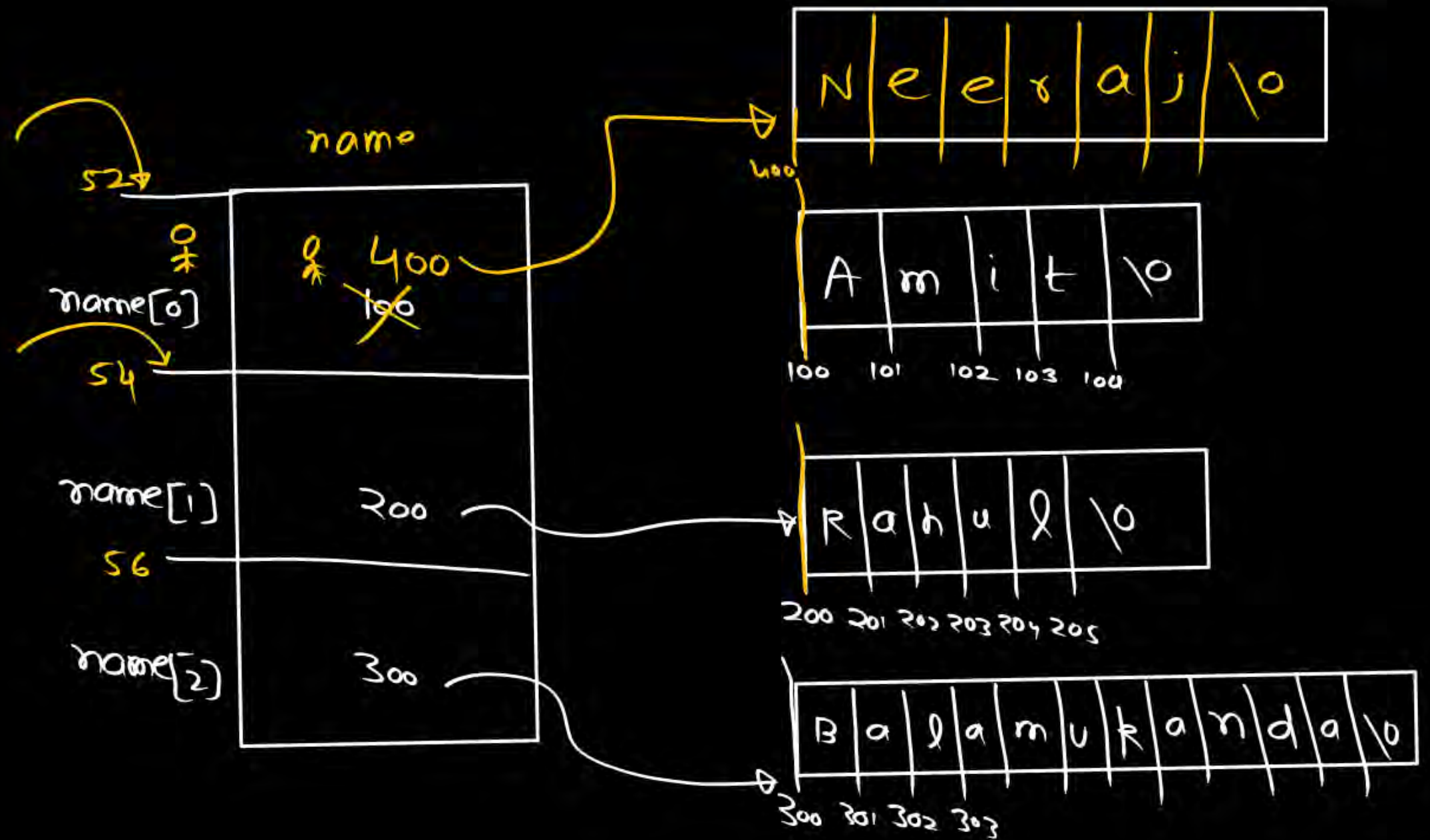
name[0][0] = 'a'; X

name



char *name[3] = {"Amit", "Rahul", "Balamukunda"};

name[0] = "Neeraj"; ✓



In-built
functions on
strings

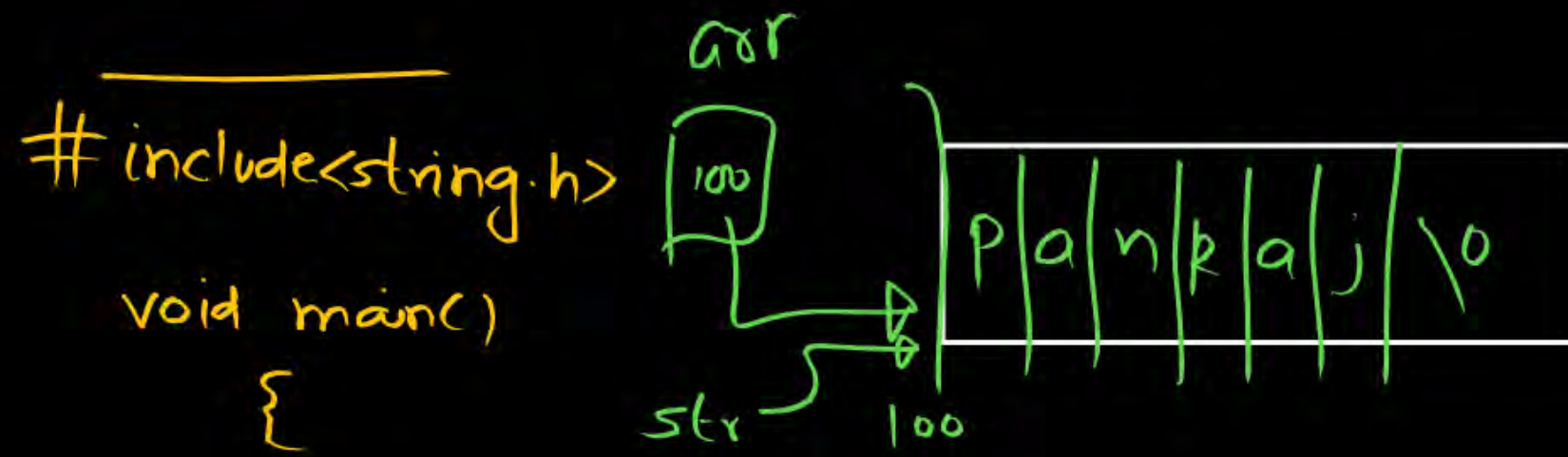
string.h

✓ {
 (i) strlen
 (ii) strcpy
 (iii) strcat
 (iv) strcmp
}

strlen

It returns the no. of symbols in the string pointed by pointer given to this function ('\0' is not counted)

```
unsigned int strlen( char *p );
```



```
char arr[] = "Pankaj";
int i;
i = strlen(arr); // call
printf("%d", i);
}
```

```
unsigned int strlen(const char *str)
{
```

char count ~~not~~

this function
can not
modify
original
string

}


```
#include <string.h>
```

```
void main()
```

```
{
```

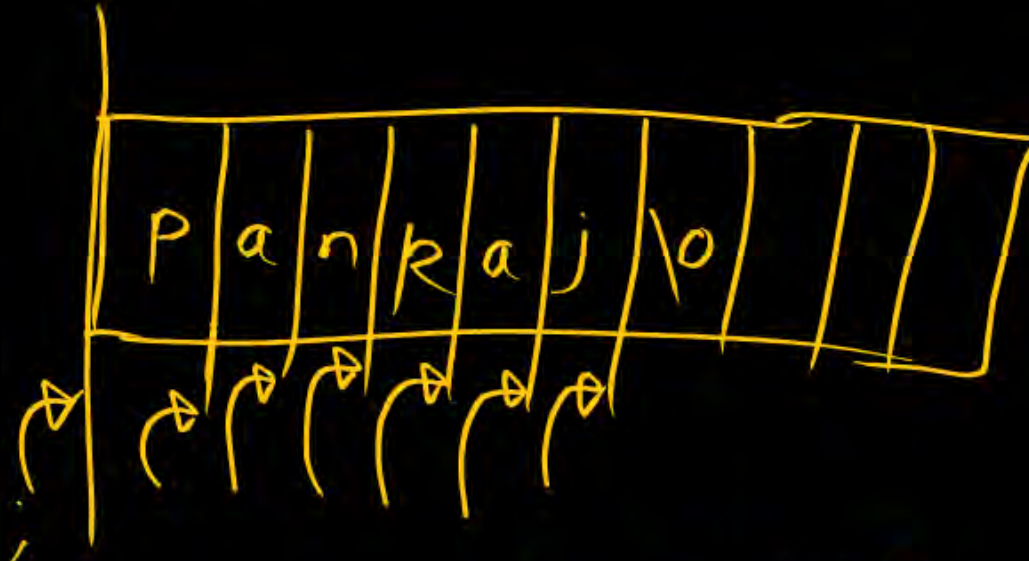
```
char arr[10] = "Pankaj";
```

```
int i;
```

```
i = strlen(arr);
```

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

```
}
```



count = 6

66

```
char *ptr = "Pankaj";
```

```
int i;
```

```
i = strlen(ptr);
```

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

6

```
char *ptr = "Pankaj";
```

```
int i;
```

```
i = strlen(ptr + 2);
```

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

4

char arr[10] = "Pankaj";

arr = "Neeraj"; Invalid

char arr[20];

arr = "Balamuponda";

Invalid

strcpy

`char * strcpy(char * destination, const char * source)`
Memory

It copies the string pointed by source pointer

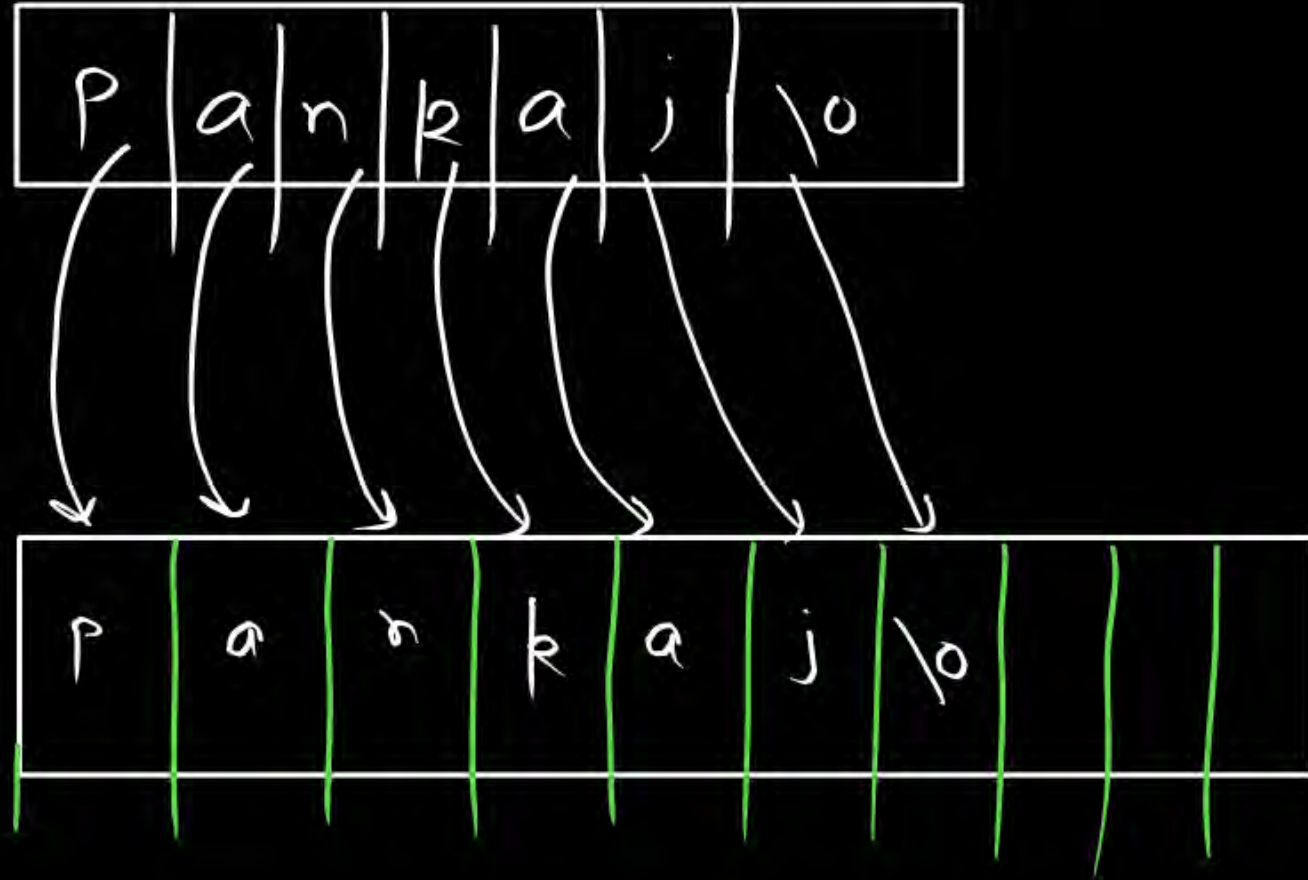
to the memory/buffer pointed
by destination pointer.

(also copy \0)

① char arr[10];

arr = "Pankaj"; Invalid

strcpy(arr, "Pankaj"); ✓
↓



②

char *ptr;

strcpy(ptr, "Pankaj");

③

char arr[4];

strcpy(arr, "Pankaj");

To

avoid overflow type error
size of dest memory must
be enough to accomodate
new string


```
char arr[10];
```

```
char *ptr = "Pankaj",
```

```
strcpy( arr , ptr )
```

```
printf("%s", arr);  $\Rightarrow$  Pankaj
```

strcat

To concatenate strings

{ It appends one string at the end of other string.

char * strcat(char * destination, const char * source)

It must
be an array

destination
pointer

It appends
string pointed by
source pointer
at the end of
string pointed by

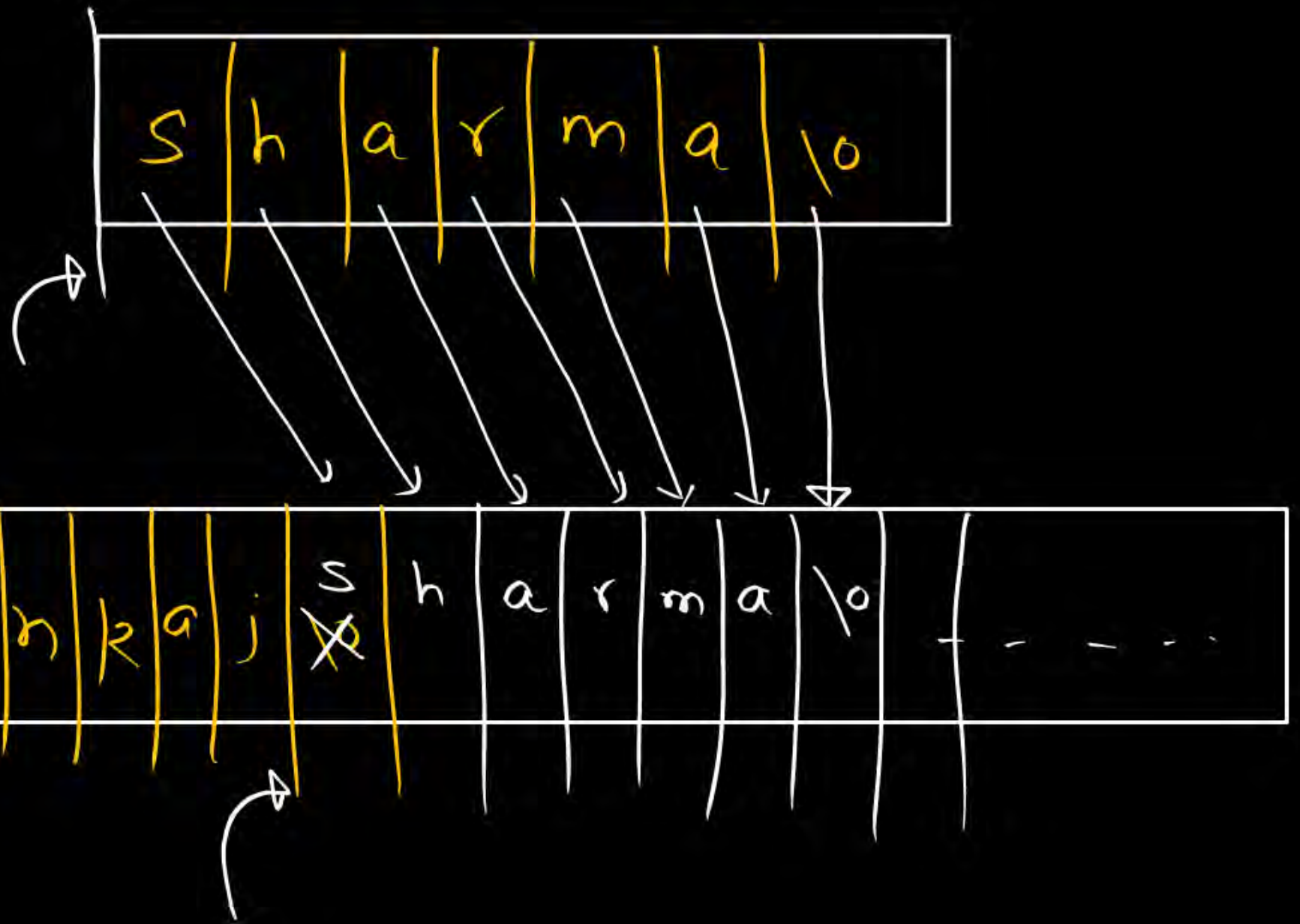
char arr[20] = "Pankaj";

char *ptr = "Sharma";

strcat(arr, ptr);

printf("%s", arr)

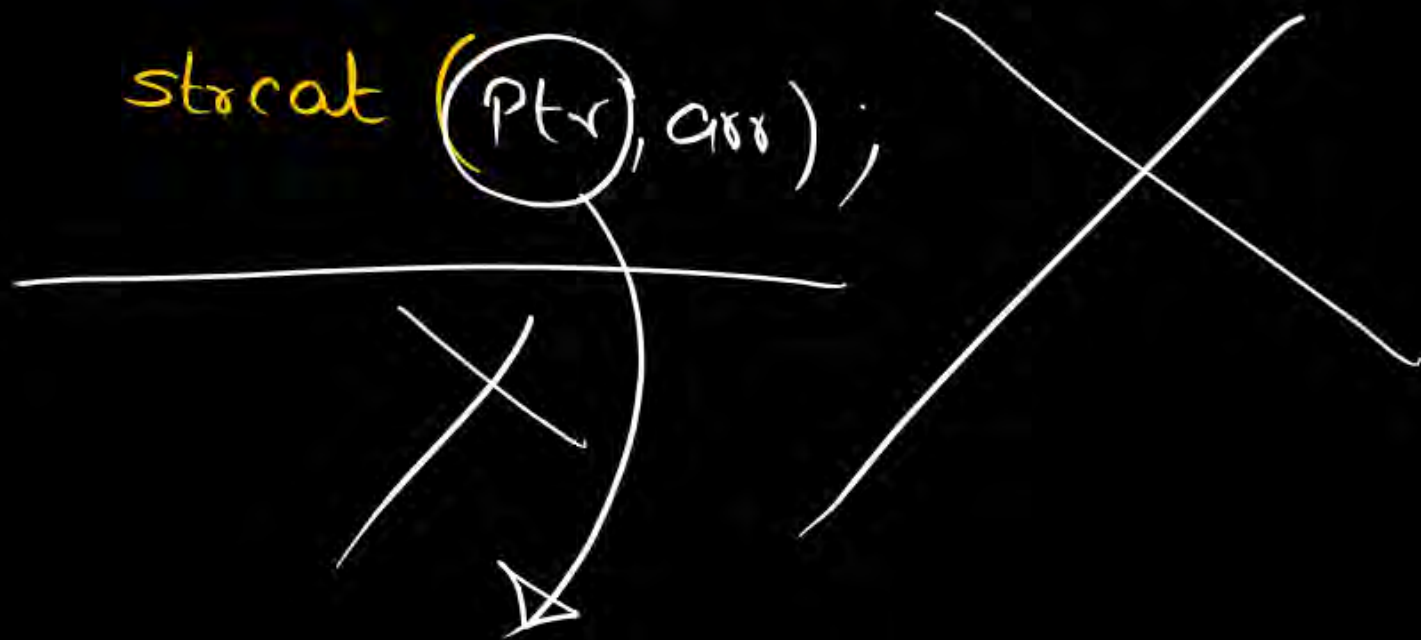
PankajSharma ✓



char arr[20] = "Pankaj";

char *ptr = "Sharma";

strcpy(ptr, arr);



H.W { strcpy
compare }

09:00 PM

↓
structure & union

