unacademy

Arrays & Pointers - Part XI

Comprehensive Course on C- Programming

Pankaj Sharma • Lesson 29 • June 7, 2024

unacademy

CS & IT Engineering

C Programming

Arrays & Pointers -XI

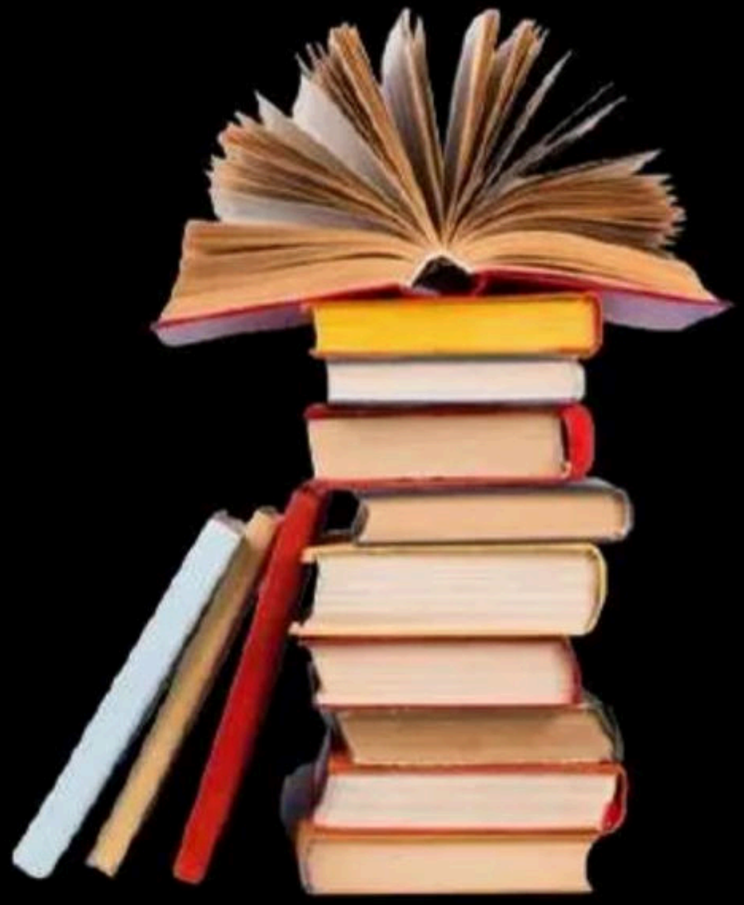
Lecture Number- 29

By- Pankaj Sir

unacademy

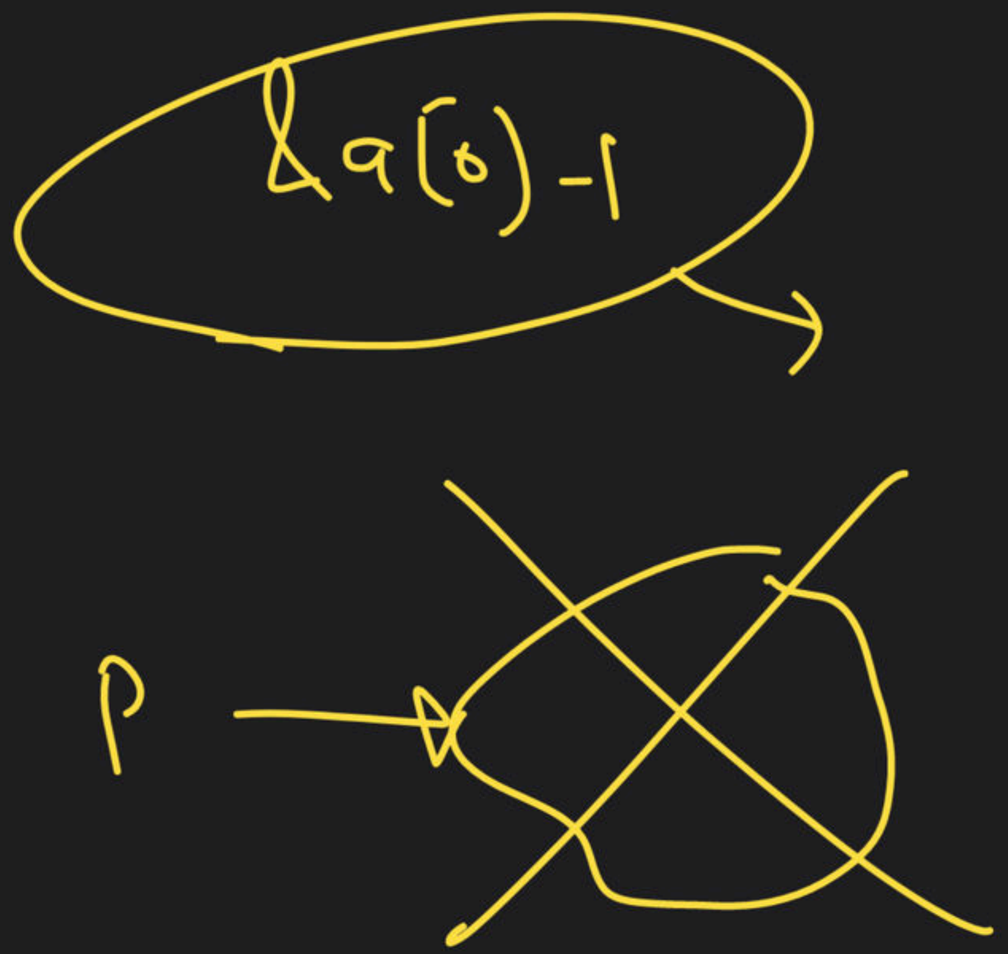
Topics

to be covered

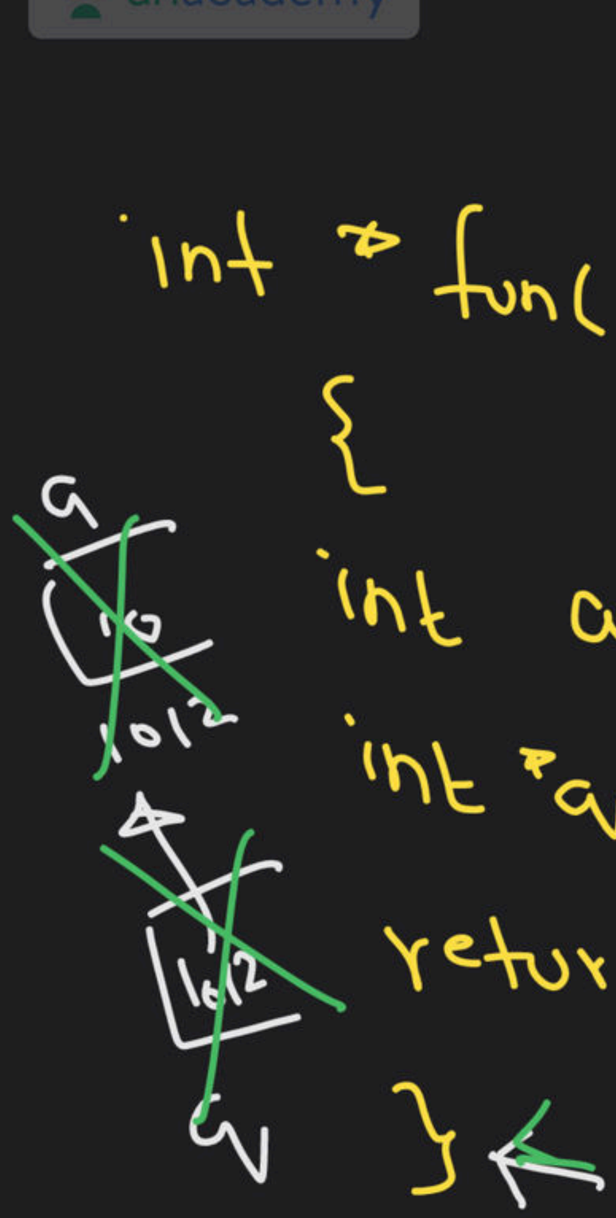


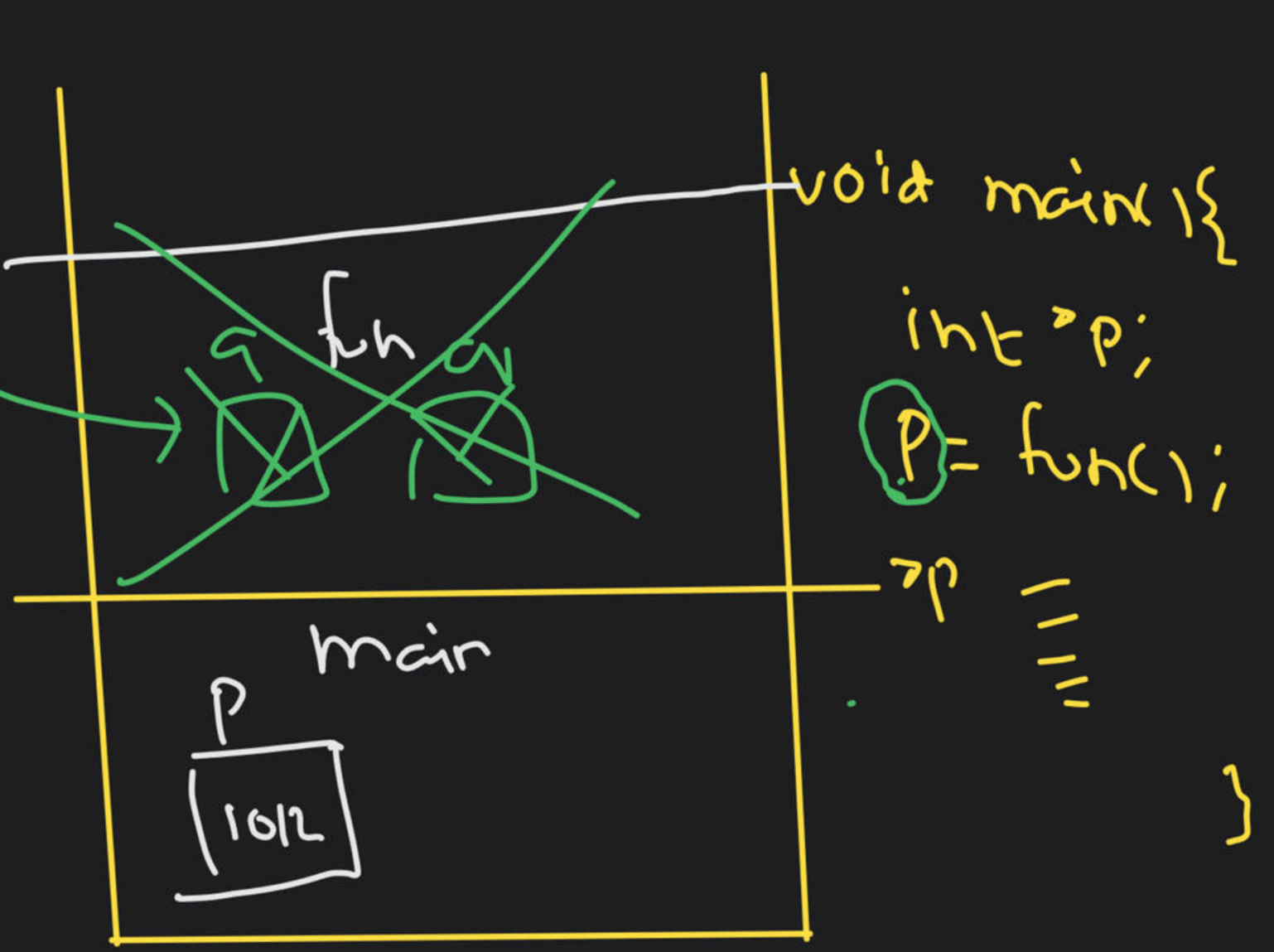
1. Arrays & Pointers-XI

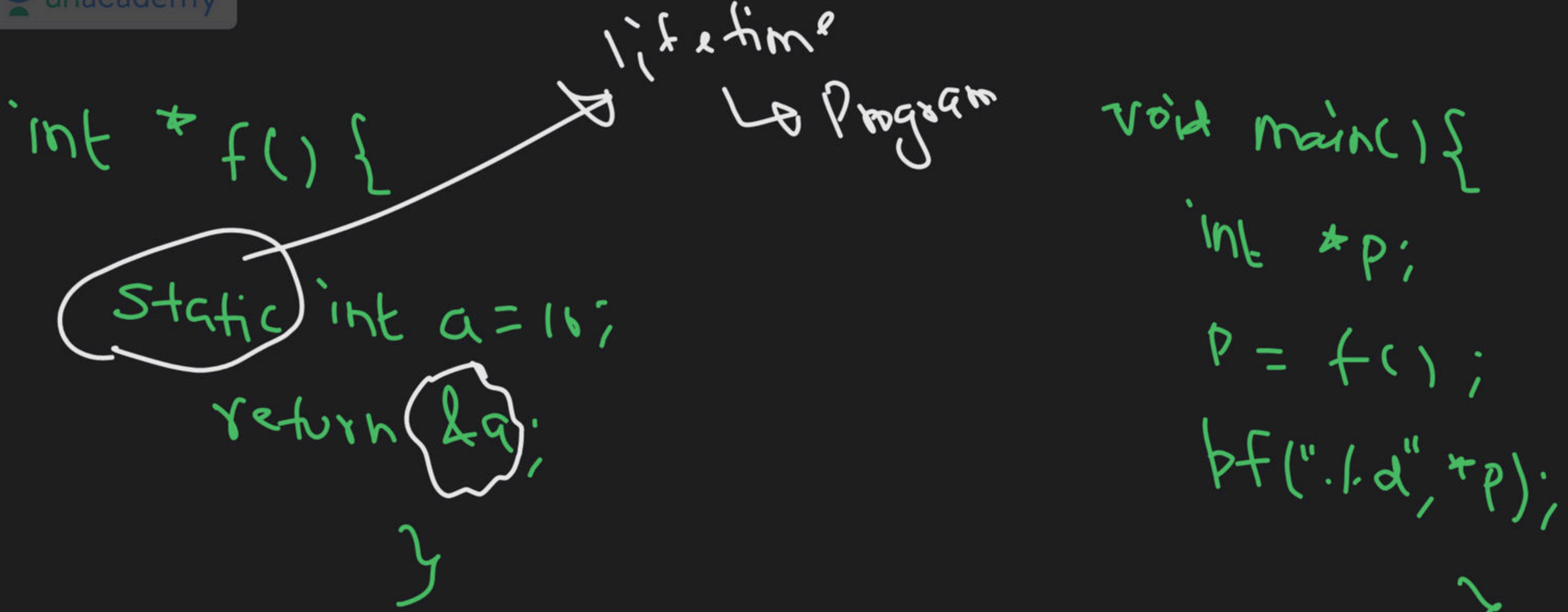
Dangling Pointer



```
int * fun()
{
    int a = 10;
    int *q = &a;
    return q;
}
```






int * f() {
lifetime
Lo Program
static int a = 10;
return &a;
}
void main() {
int *p;
p = f();
bf("%d", *p);
}

NULL pointer

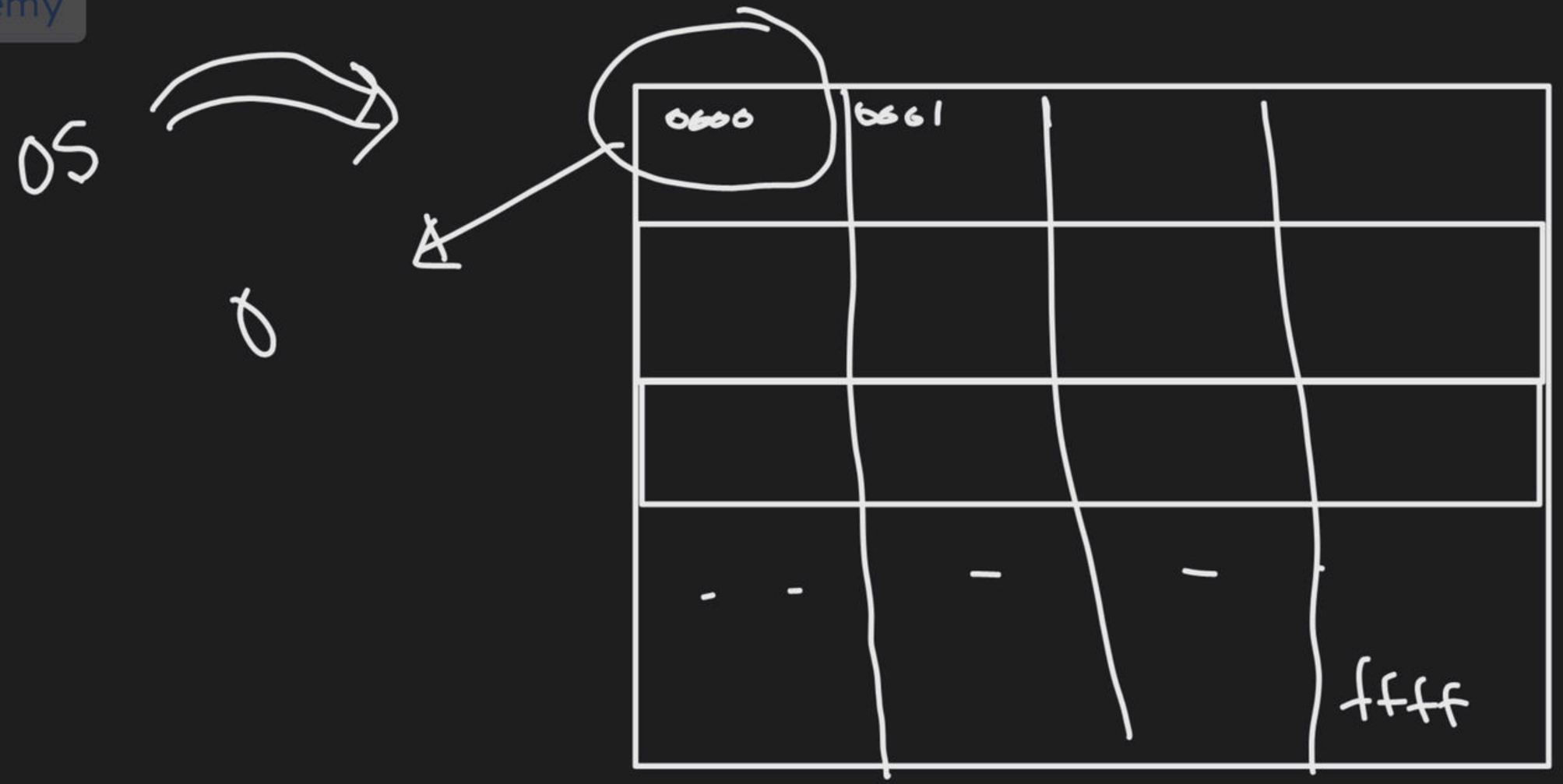
Specially designed pointer

- 112 ✗

-112.3 ✗

(i) -ve →

(ii) zero →



```
if (0) {

}
```

```
int *p = (int*)0;
```

```
ptr == NULL

if (ptr){
    ========
}
```

Dynamic Memory Allocation

`int a[5000];`

→ 400

waste

`int a[400];`

→ 5000

DMA

1. malloc()
2. calloc()
3. realloc()
4. free()

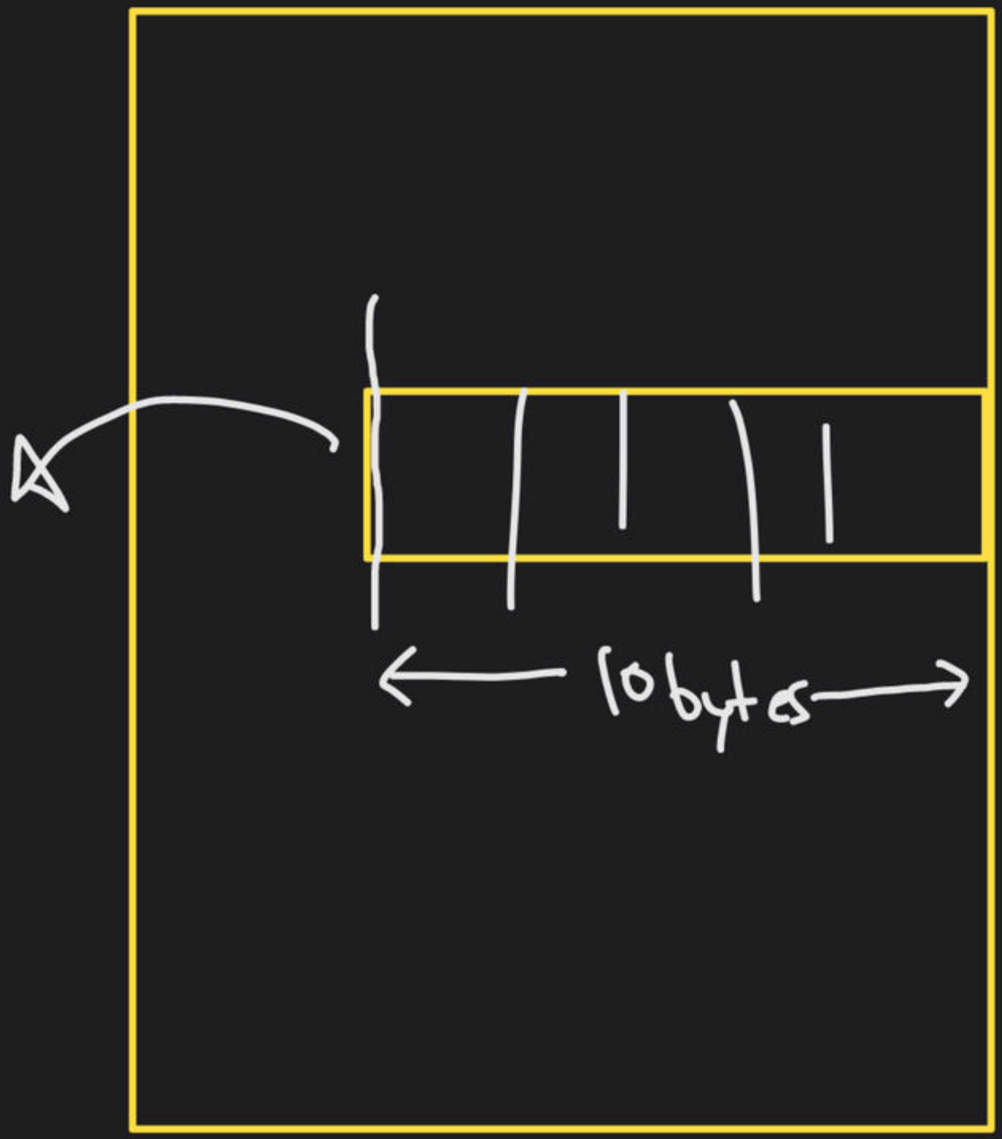
unacademy

malloc

-ve X

malloc(size in bytes)

(void *) malloc(10);



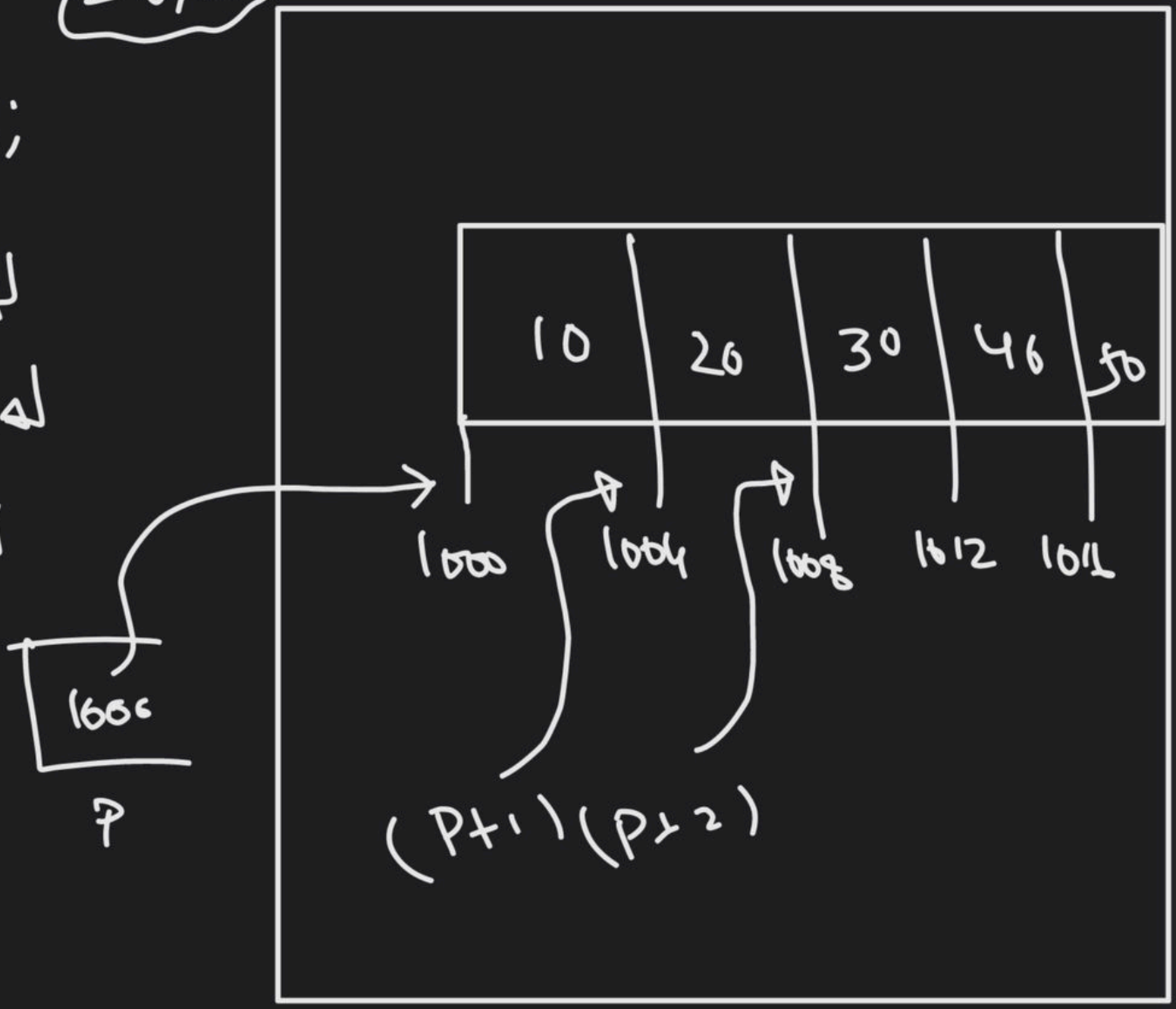
size_t

```
(void *)malloc(unsigned int);
```

unacademy

2 byte

```
int *p = malloc(10);
scanf("%d", p);      10↵
scanf("%d", p+1);    20↵
scanf("%d", p+2);    30↵
scanf("%d", p+3);    40↵
scanf("%d", p+4);    50↵
```

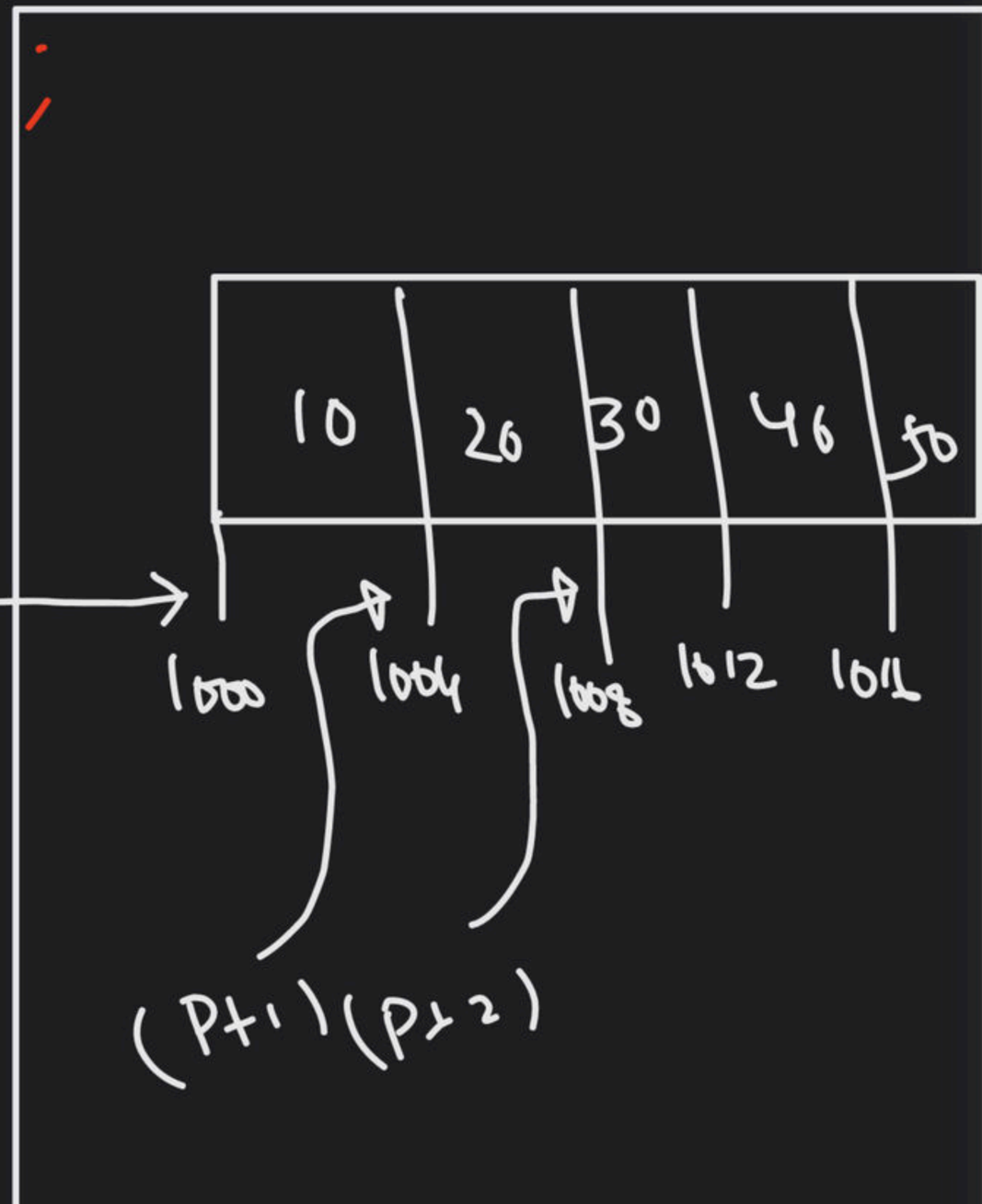


unacademy

```
int *p = malloc (5 x sizeof(int));
```

```
scanf("%d", p+0);
scanf("%d", p+1);
scanf("%d", p+2);
scanf("%d", p+3);
scanf("%d", p+4);
```

10 ✓
20 ✓
30 ✓
40 ✓
50 ✓



unacademy

```
int *p = malloc (5 x sizeof(int));

scanf("%d", p+0);
scanf("%d", p+1);
scanf("%d", p+2);
scanf("%d", p+3);
scanf("%d", p+4);

for(i=0; i<5; i++)
    scanf("%d", p+i);
```

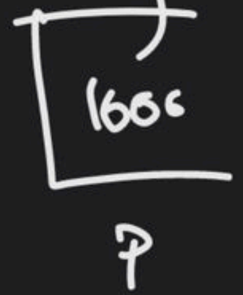
100

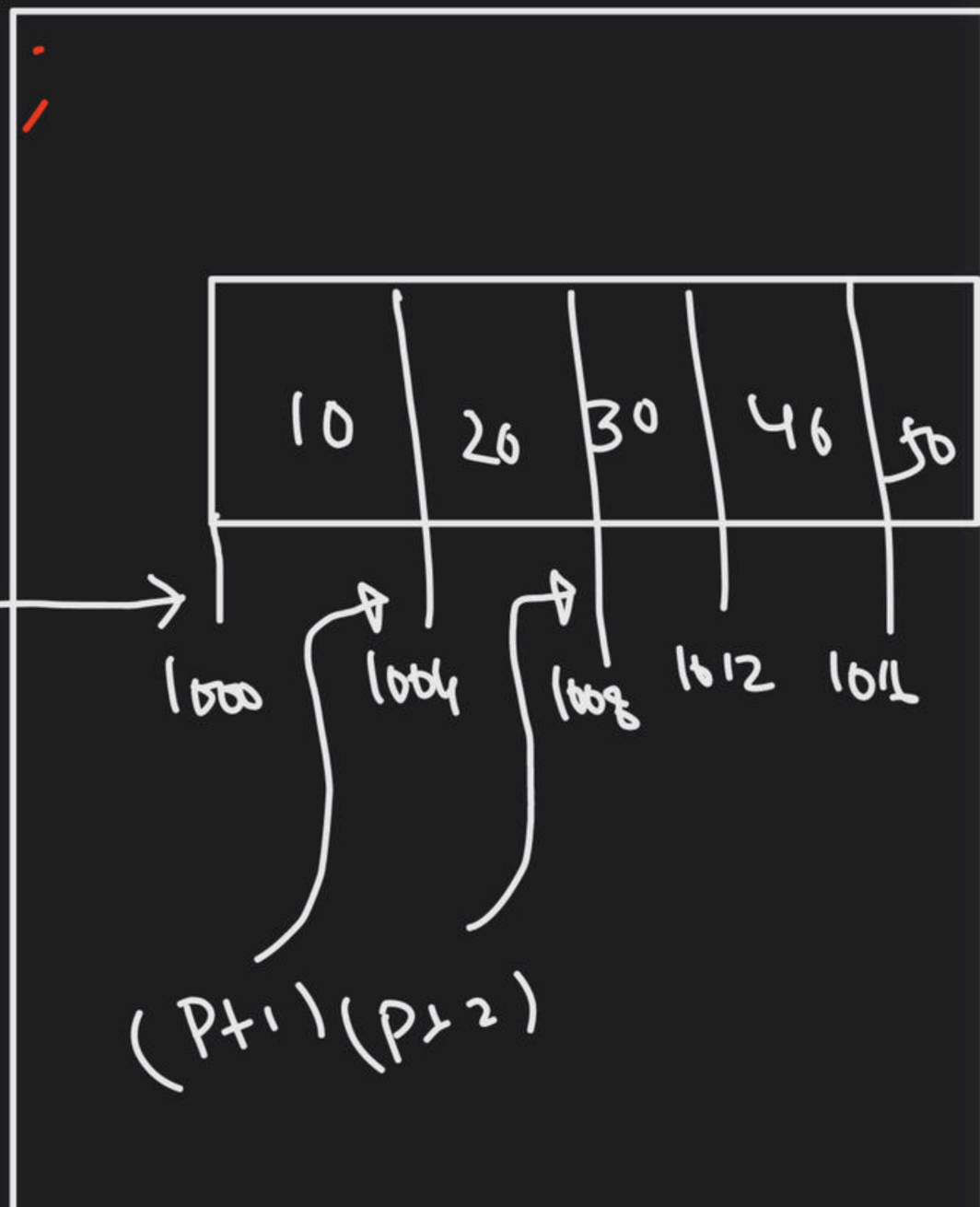
200

300

400

500





unacademy

```
int *p = malloc (5 x sizeof(int));
```

```
for(i=0; i<5; i++)
    scanf("%d", p+i);
```

5 values read

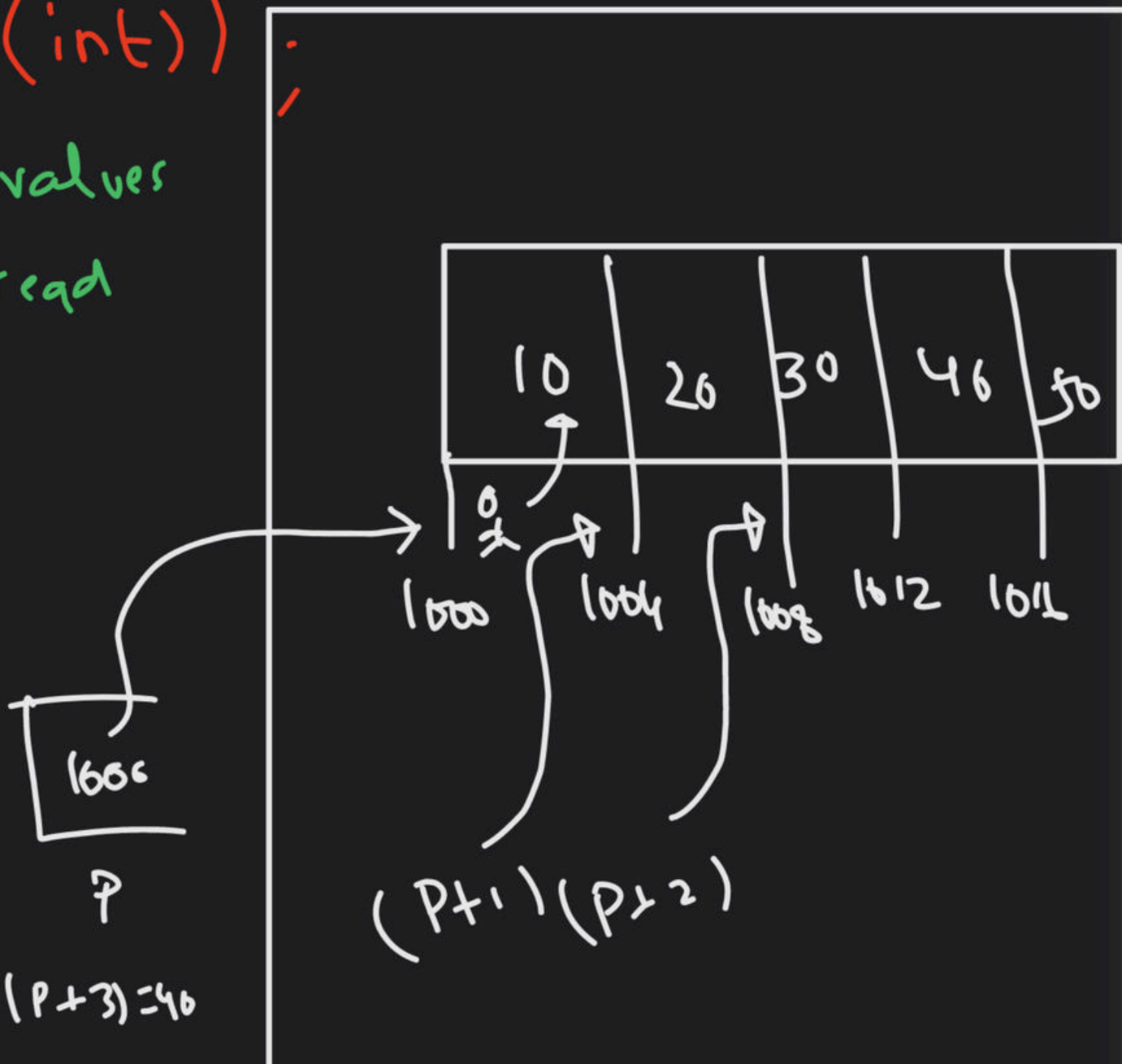
p ⇒ Memory loc. 1000

(p+0) →

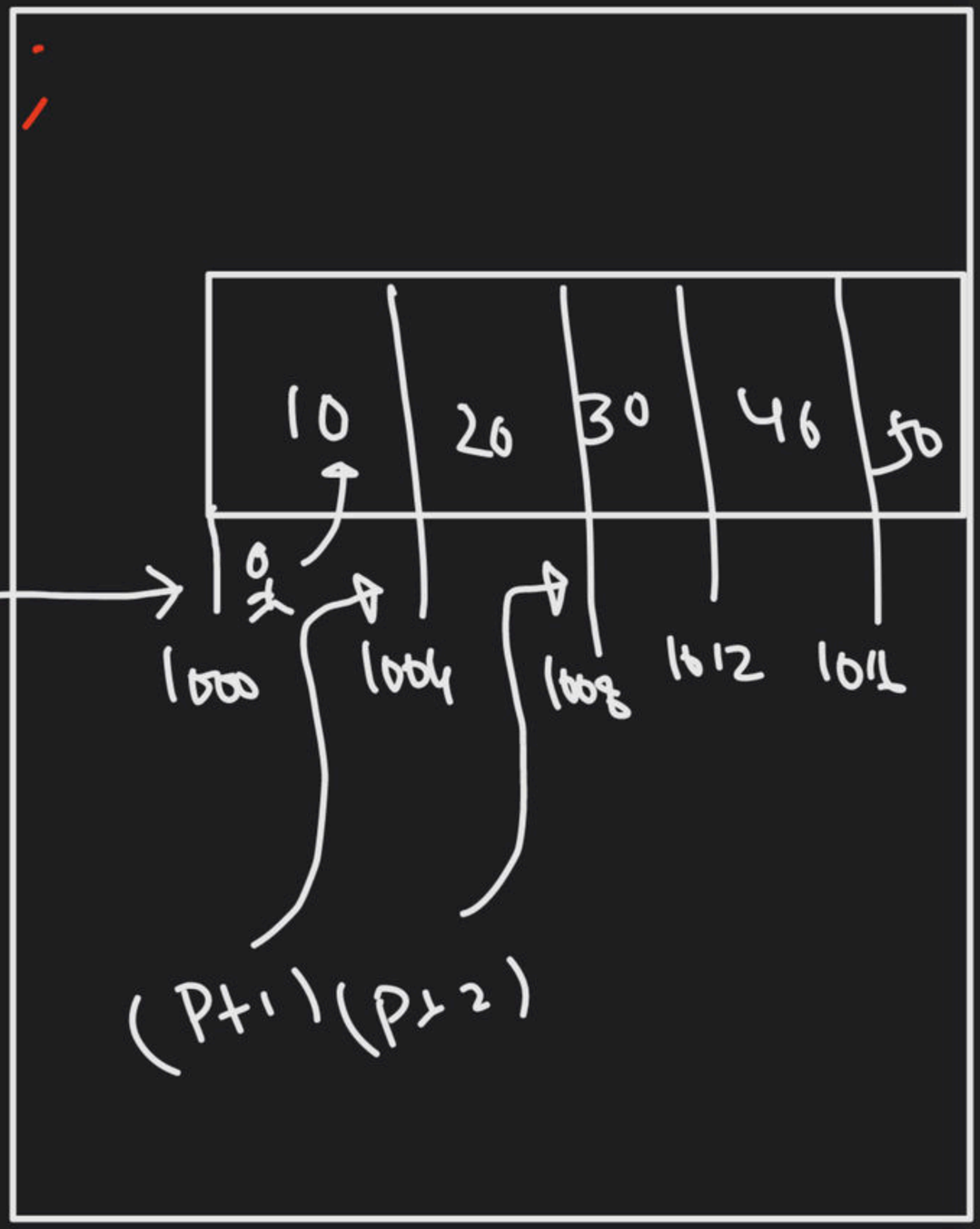
*(p+0) ⇒ 10

(p+1) ⇒ Memory loc. 1004

*(p+1) ⇒ 20, *(p+2) = 30, *(p+3) = 40



```
int *p = malloc (5 x sizeof(int));

for(i=0; i<5; i++)
    scanf("%d", p+i);          } 5 values read

pf("%d", *(p+0));   10
pf("%d", *(p+1));   20
pf("%d", *(p+2));   30
pf("%d", *(p+3));   40
pf("%d", *(p+4));   50
```



unacademy

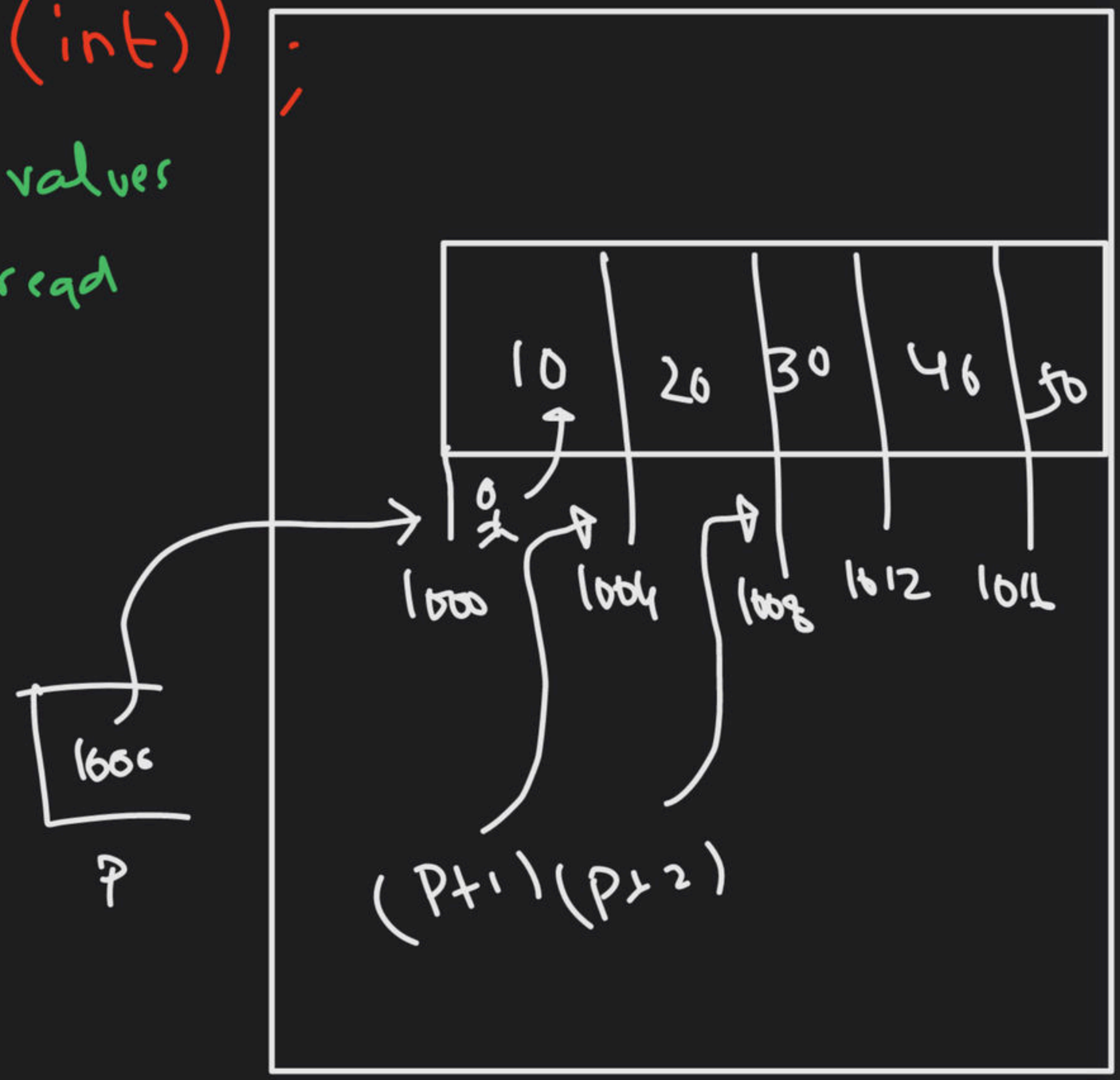
```
bf("%d", *(p+0));
bf("%d", *(p+1));
bf("%d", *(p+2));
bf("%d", *(p+3));
bf("%d", *(p+4));
```

⇒

```
for(i=0; i<5; i++)
    bf("%d", *(p+i));
```

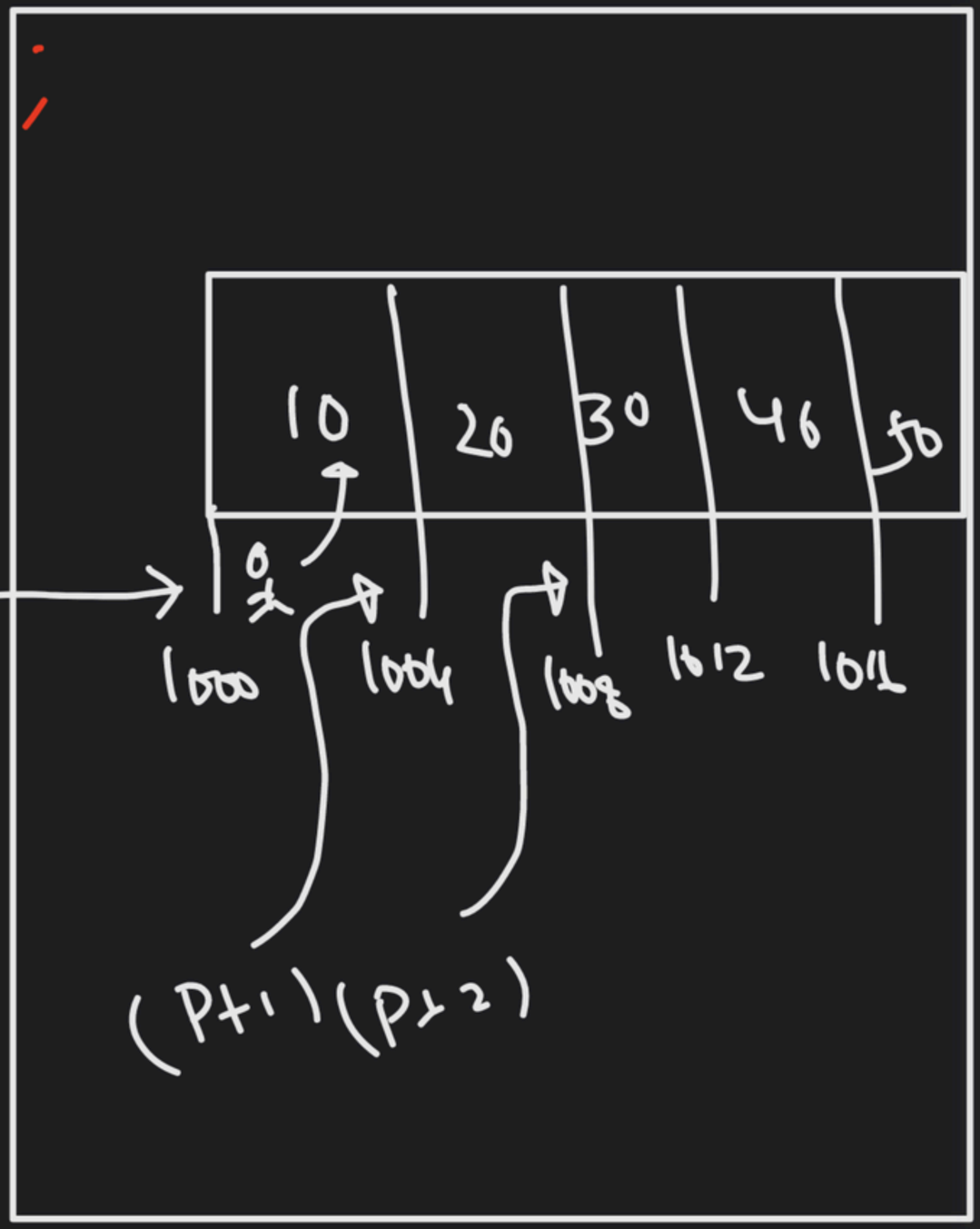
```
int *p = malloc (5 x sizeof(int));

for(i=0; i<5; i++)
    scanf("%d", p+i);

for(i=0; i<5; i++){
    printf("%d", *(p+i));
}
```

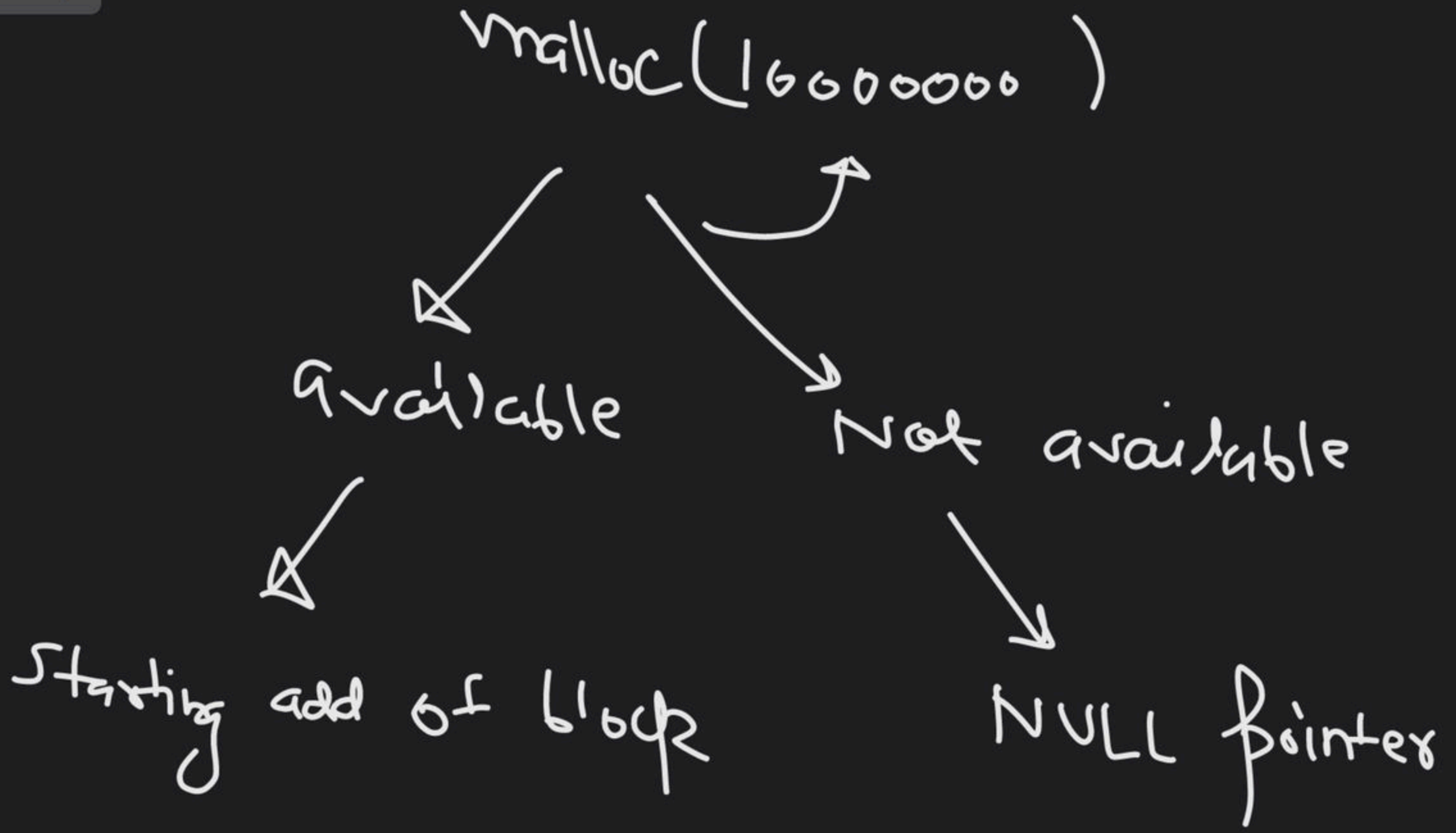
5 values read



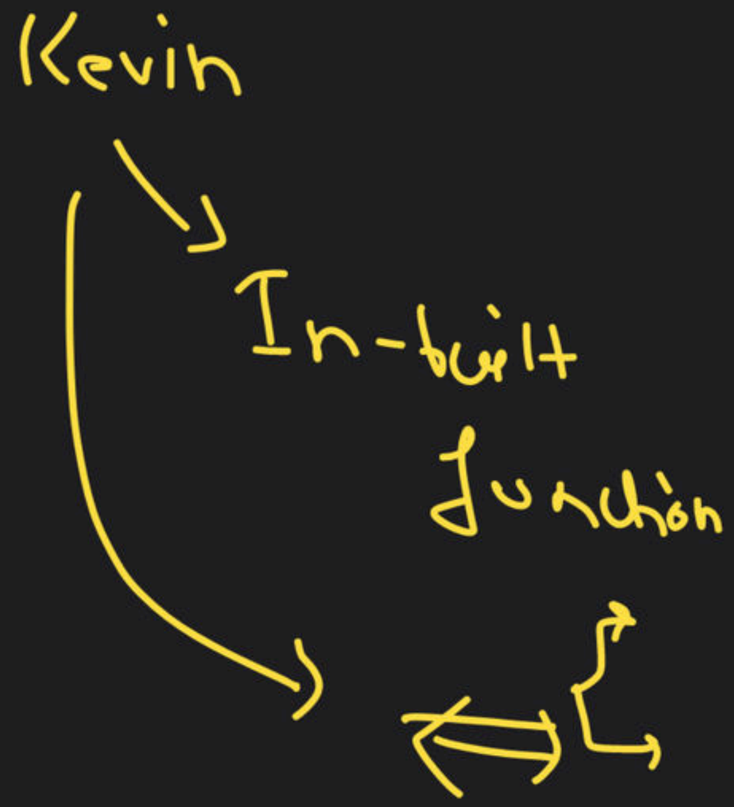
```
int *p = malloc (5 x sizeof(int));

for(i=0; i<5; i++)
    scanf("%d", p+i);

for(i=0; i<5; i++)
    bf("%d", p[i]);
}
```

5 values read




malloc(16000000)
available
Not available
Starting add of block
NULL pointer

```
int N, i;
int *p;
printf("Enter the no. of element");
```




```
if (ptr == NULL)
{
    printf("This is Kevin
            from world
            of Programming");
```

```
int N, i ;
int *p;
printf(" Enter the no. of element(");
scanf("%d", &N);
P = malloc(sizeof(int) x N);
if (P != NULL){
  for(i=0; i<N; i++)
        scanf("%d", P+i);
   for(i=0; i<N; i++)
     }     pf("%d", P[i]);
```

unacademy

Calloc

2 argument

calloc (No. of block , size of each block)

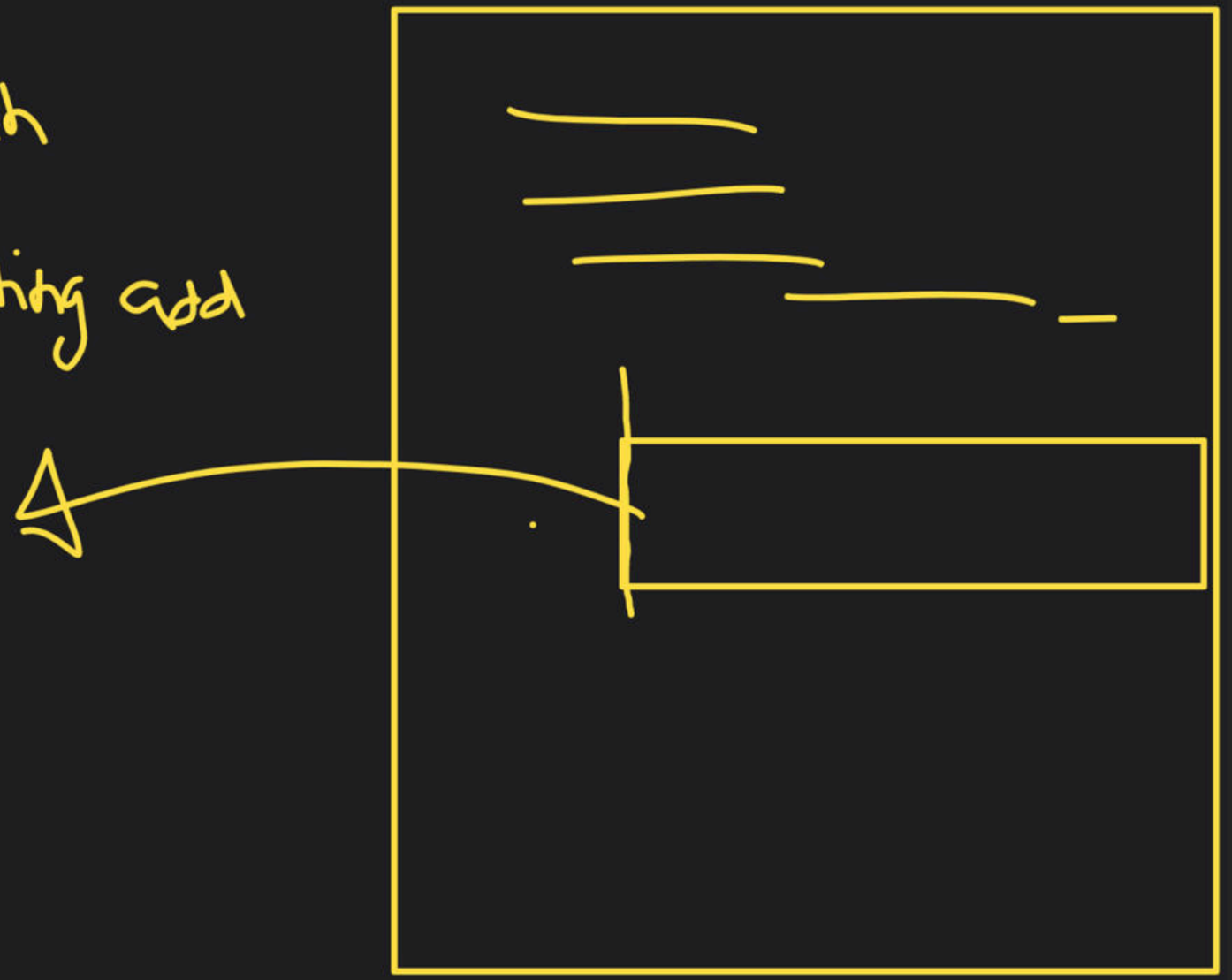
malloc(5 x sizeof(int)) ⇒ calloc (5, sizeof(int));

unacademy

Diff.

malloc ( ) → ① Search

② Starting add



unacademy

calloc

1. Search
2. 0 → bits (initialized)
3. starting add. return

more time

Expensive

Reliable

malloc : cheaper

Not reliable

unacademy

realloc

Again

used

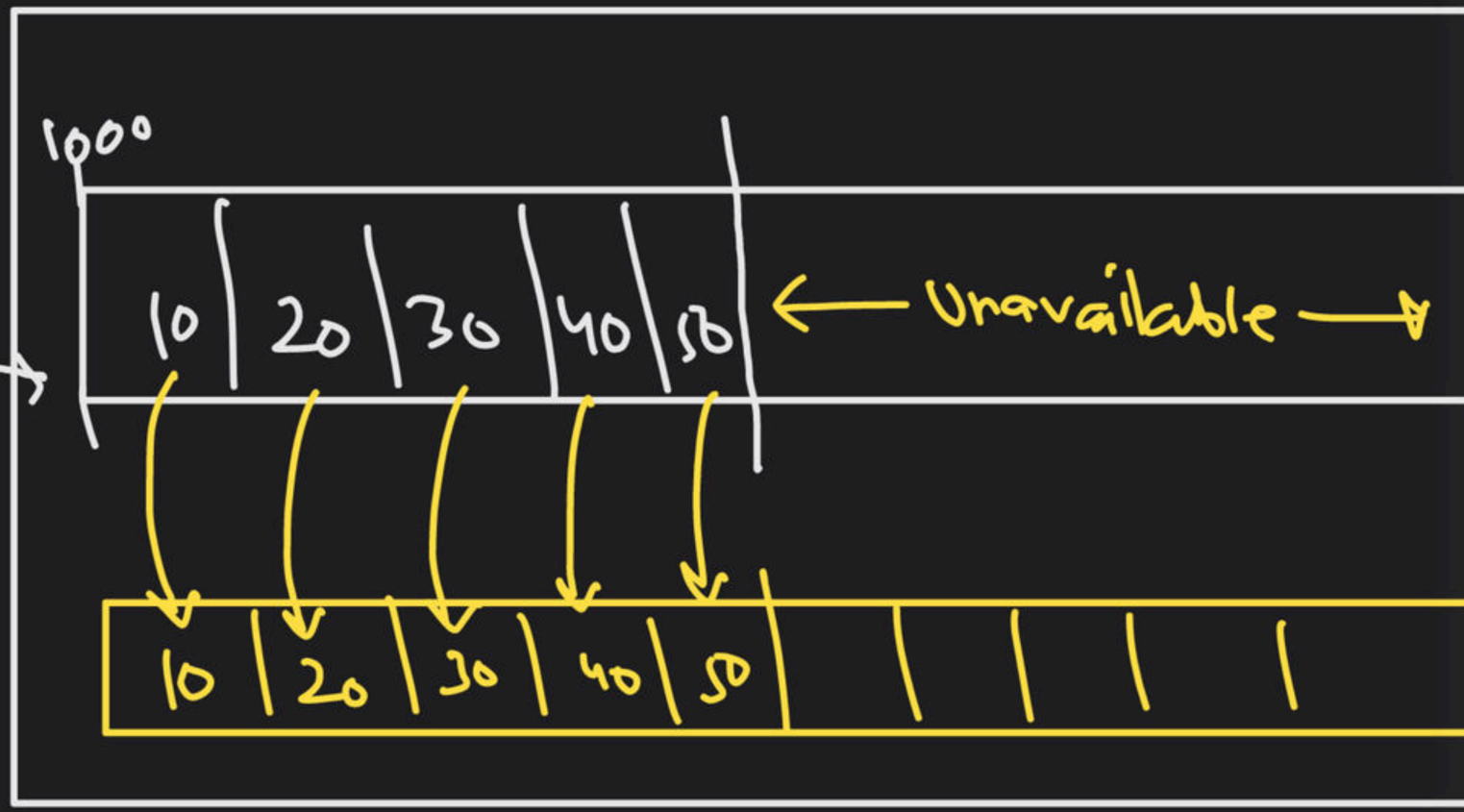
when memory is already allocated through some pointer by using malloc or calloc)

```
int *p = malloc(5 x sizeof(int));
```

for —

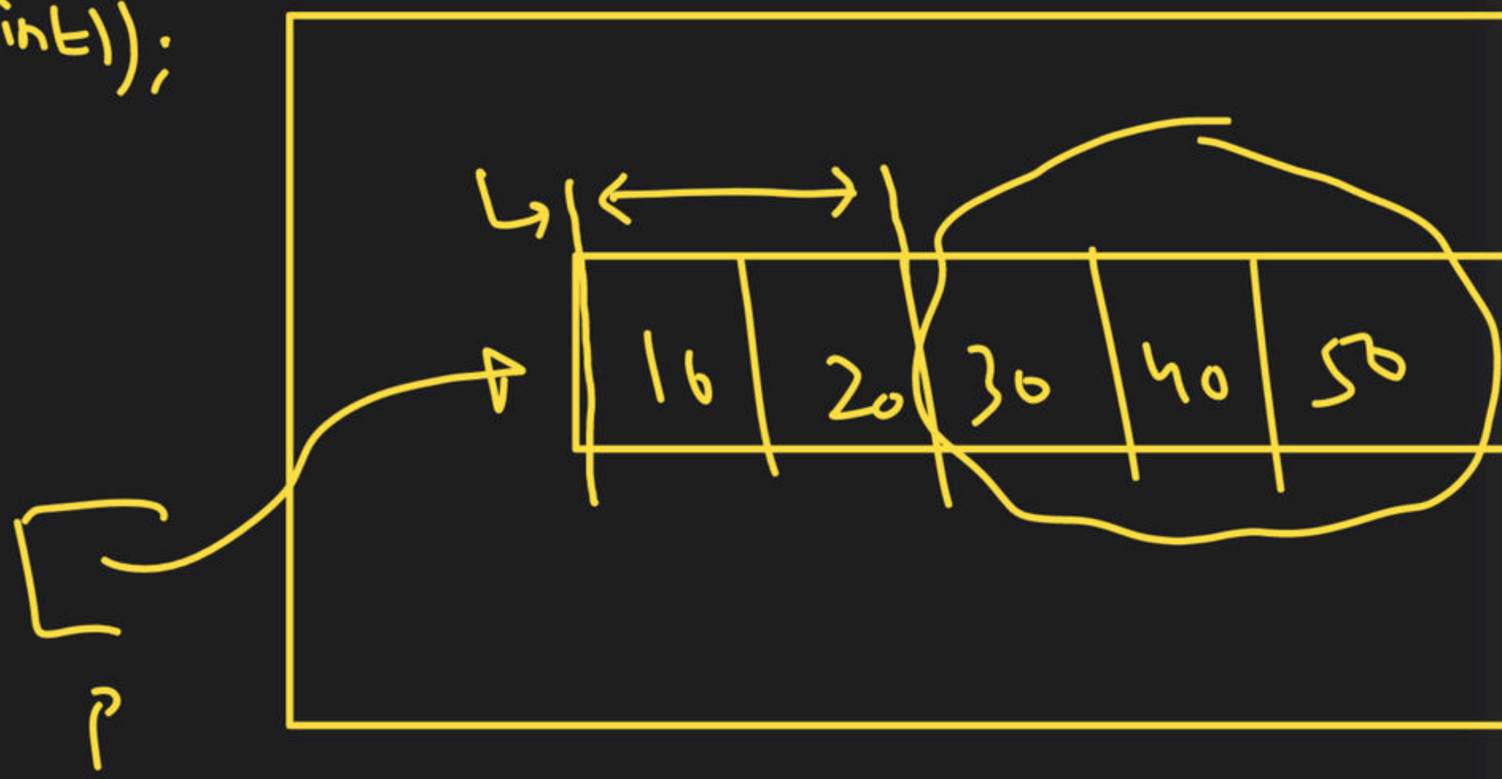
```
p = realloc(p, new size)
```

```
p = realloc(p, 10 x sizeof(int));
```



unacademy

```
int * p = malloc ( 5 * sizeof(int));
for(        ——— )
=
p = realloc (p, 2*sizeof(int));
```



DMA

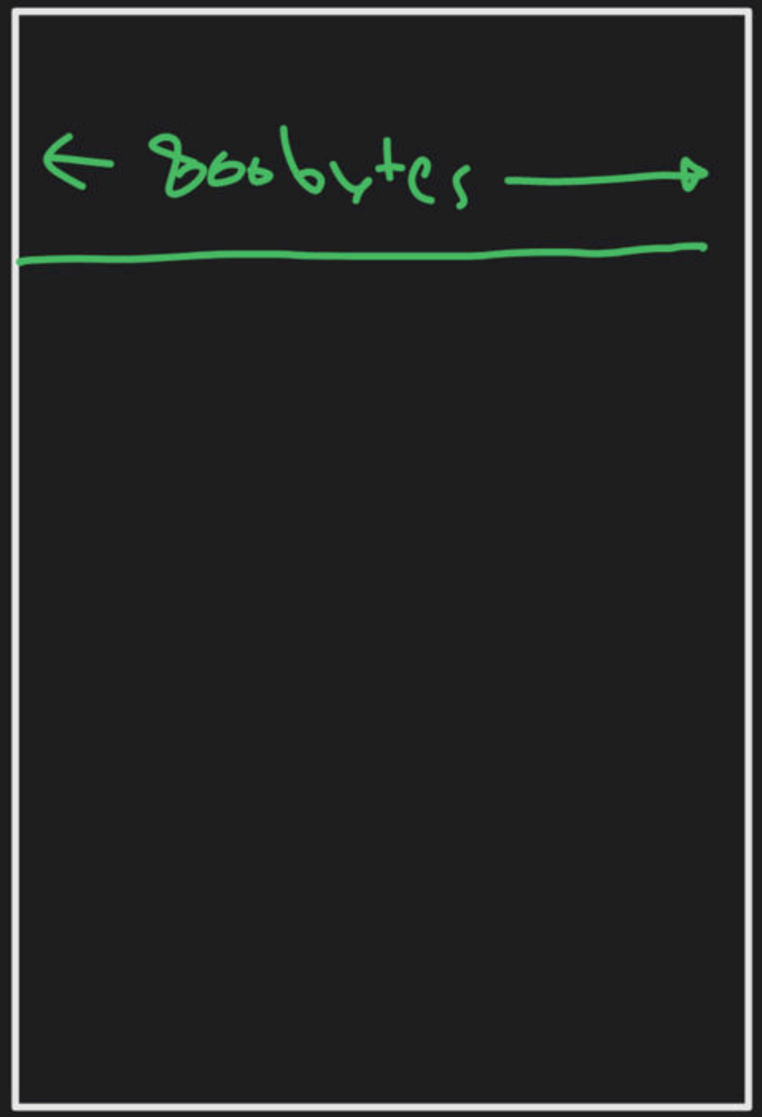
* Program

Programmer → allocate
Programmer → de-allocate

```
void fun(){
int *p = malloc(200*sizeof(int));
```

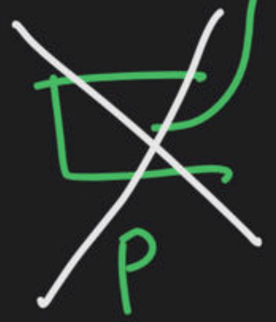
local variable

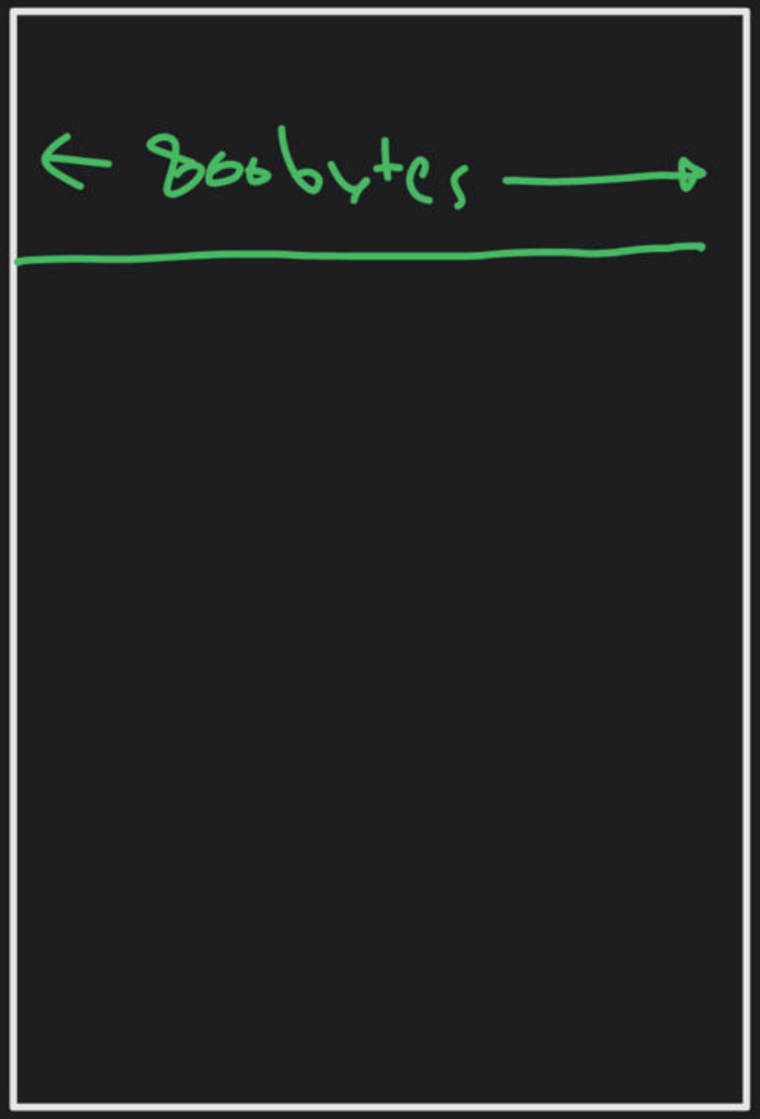
```
return;
}
void main(){
fun();
fun();
fun();
fun(); fun();}
```



```
void fun(){
int *p = malloc(200*sizeof(int));
return;
}
void main(){
fun();
fun();
fun();
fun();fun();}
```

local variable





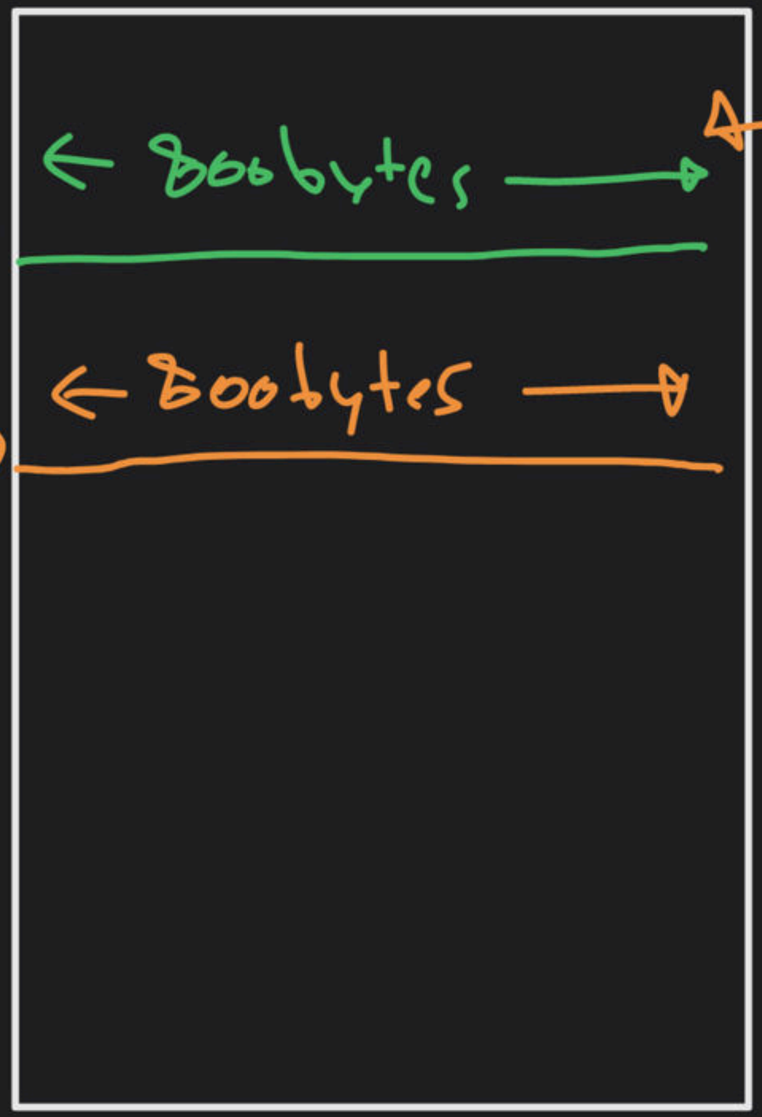
```
void fun(){
int *p = malloc(200*sizeof(int));
```

local variable

```
return;
}
```

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

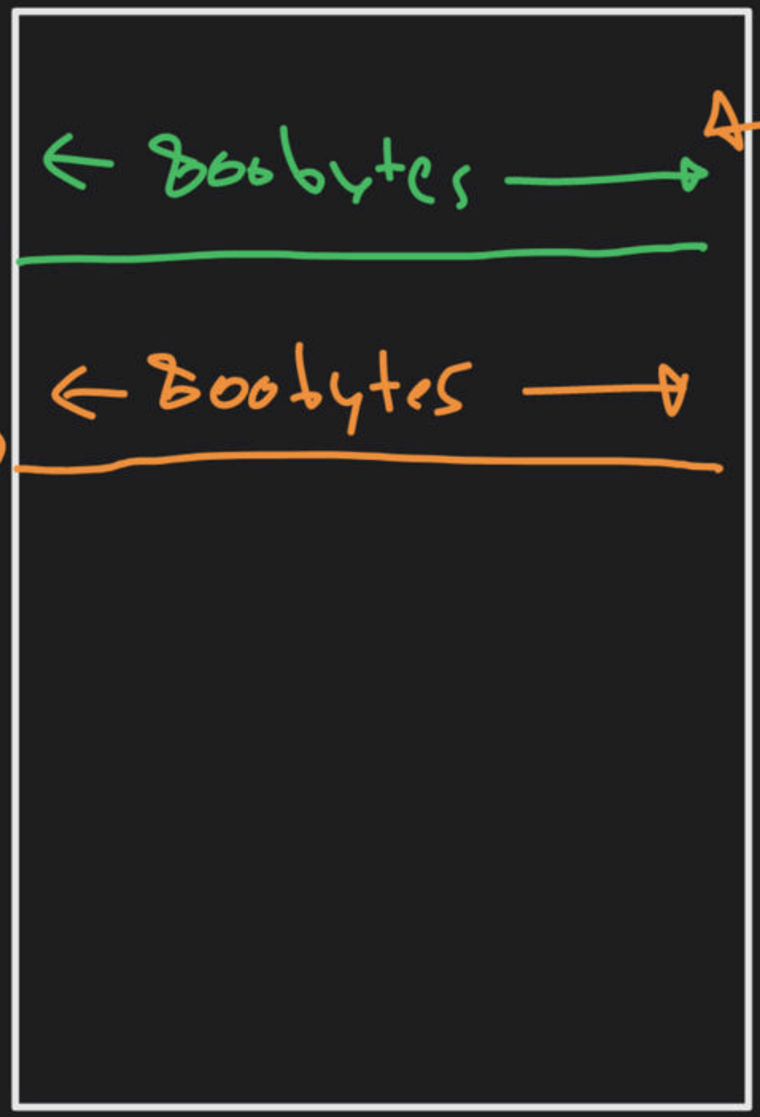
Is there any way to access this memory



```
void fun(){
    int *p = malloc(200*sizeof(int));
}
```

local variable

Is there any way to access this memory




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

return;

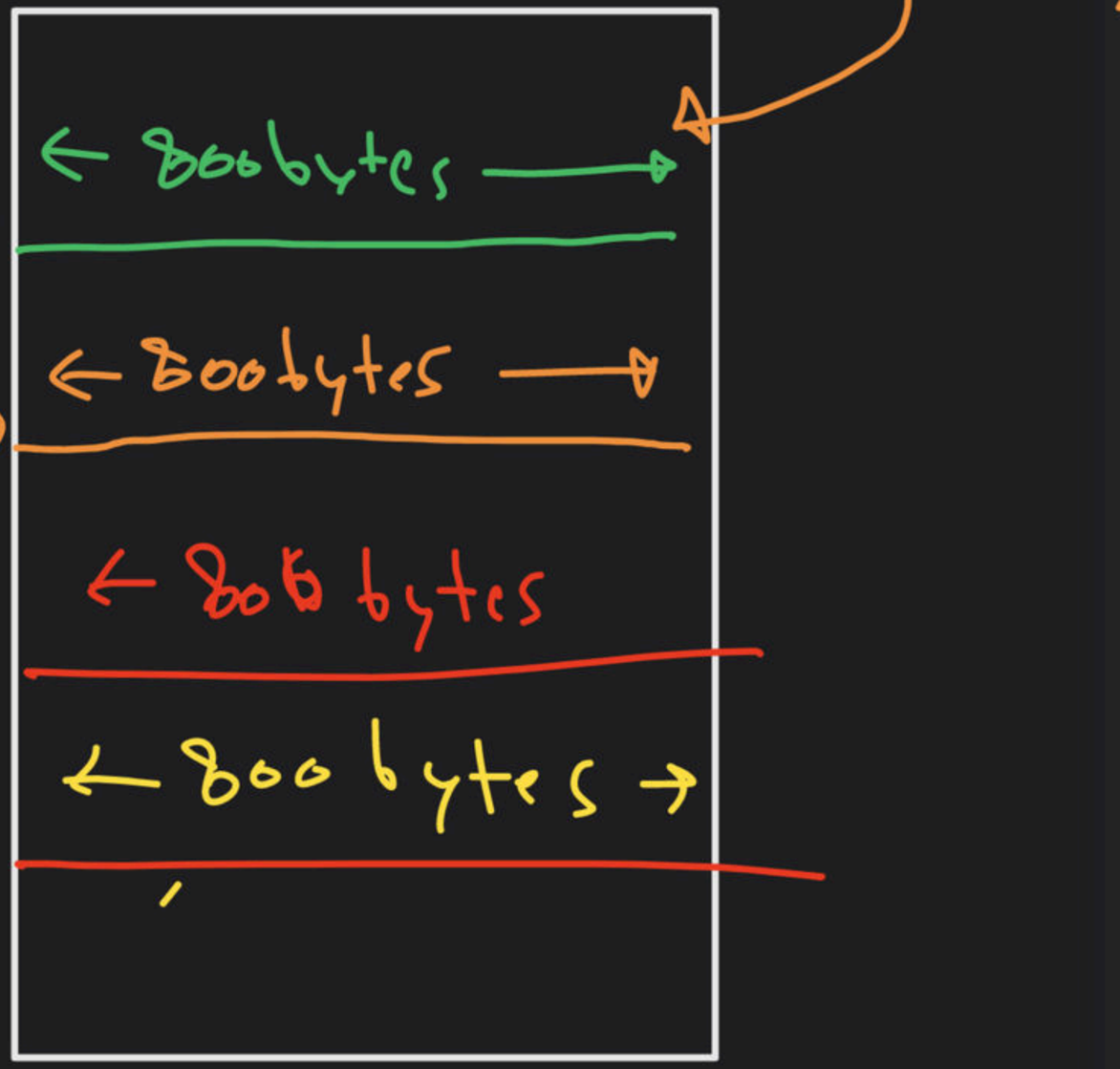
p

Is there any way to access this

```
void fun(){
int *p = malloc(200*sizeof(int));
```

local variable

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

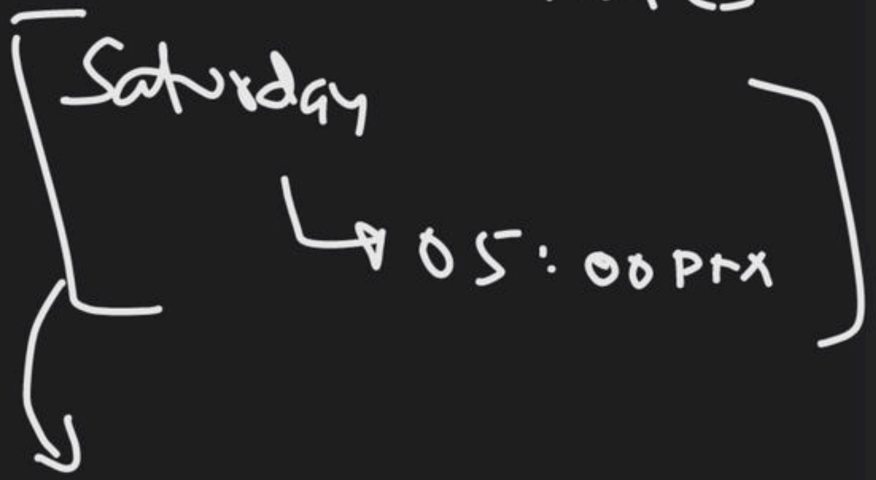


unacademy

free()

50 Questions
↳ 3 minutes

```
void fun() {
    int *p = malloc( — );
    ==
    free(p);
    return ;
}
```



String → 3-4 class

structure, union,

misc.

20th June

unacademy

▲ 1 • Asked by Vishal

sir m ye puch raha tha


```c
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>

int main() {
    // Write C code here
    int*p = malloc(2 * sizeof(int));
    p[0] = 1;
    p[1] = 2;
    printf("%d", p[0]);
    printf("%d", p[1]);
    p = realloc(p, 1*sizeof(int));
    printf("%d", p[0]);
    printf("%d", p[1]);
    return 0;
}
```

```
/tmp/28Lm1WJ1TB.o
1212

=== Code Execution Successful ===
```

unacademy

unacademy

special class

↓

5 PM

THANK YOU!

Here's to a cracking journey ahead!