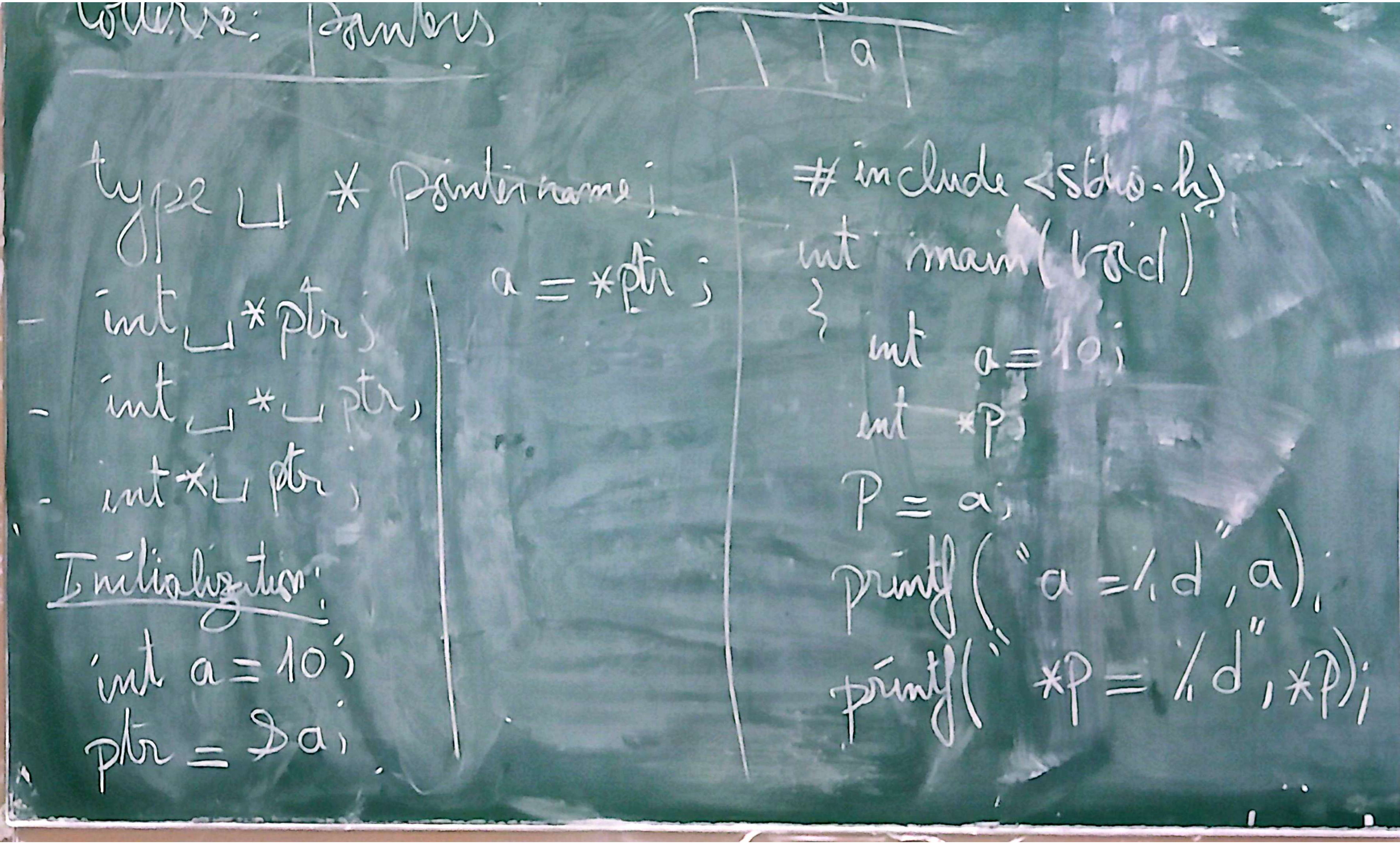


Course: Pointers

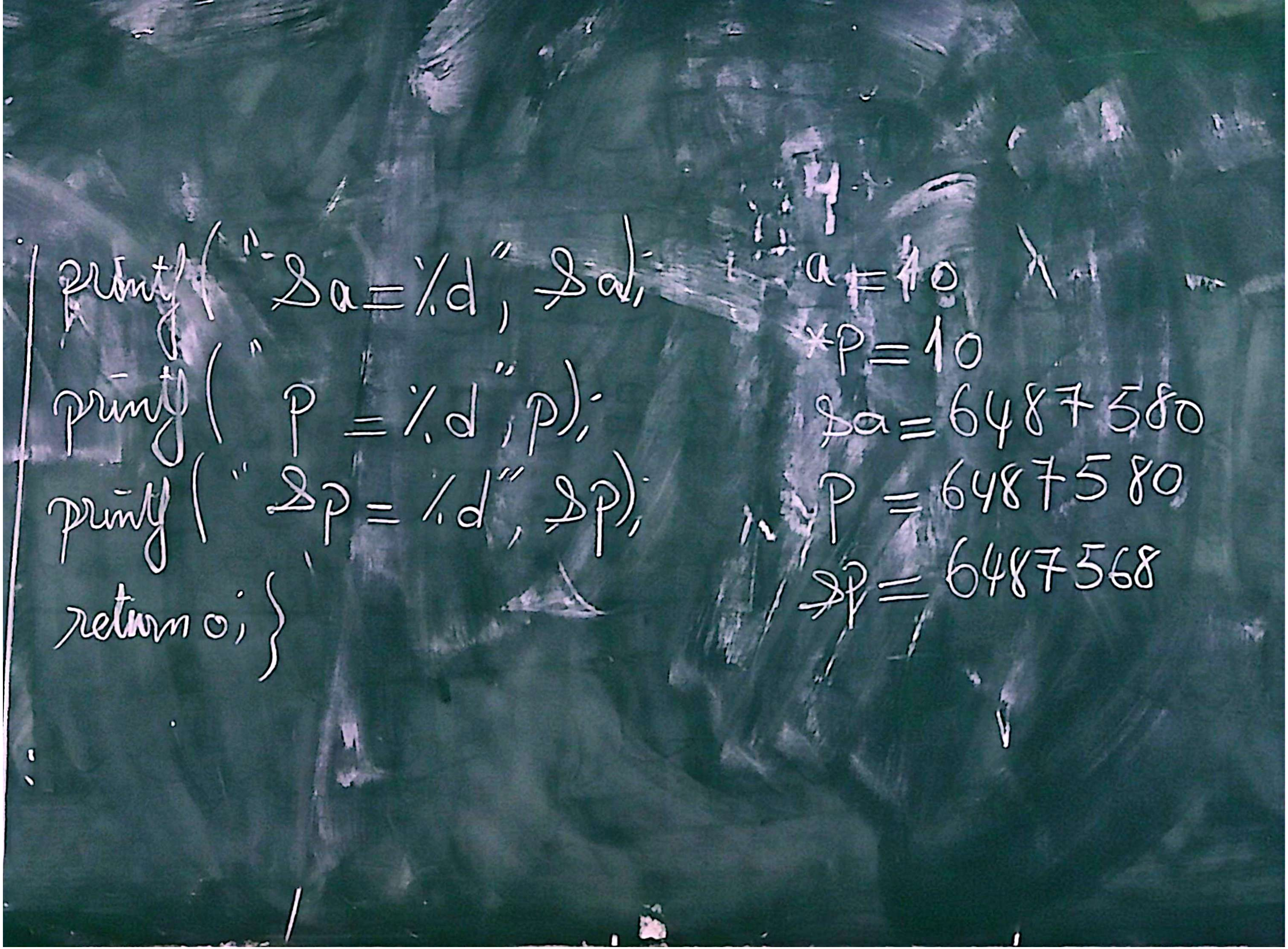
```
type ␣ * pointername;

- int␣*ptr;
- int␣*␣ptr;
- int*␣ptr;
```

Initialization:

```
int a=10;
ptr = &a;
```

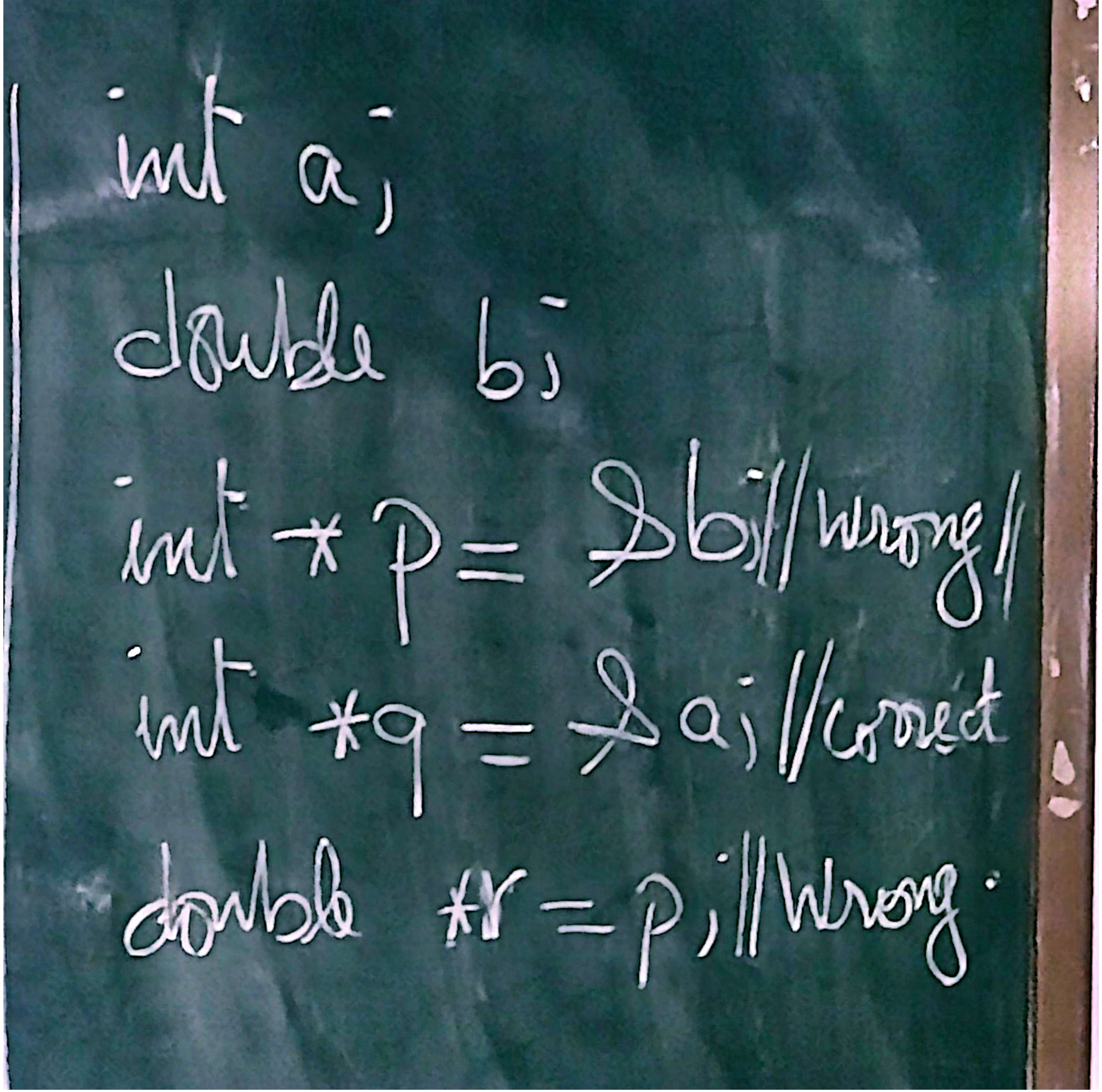
```
a = *ptr;
```

```
#include <stdio.h>
int main(void)
{
    int a=10;
    int *p;
    P = a;
    printf("a =%d", a);
    printf(" *P = %d", *p);
```



```
printf("&a=%d",&a);
printf(" p =%d",p);
printf("&p=%d",&p);
return 0;}
```

```
a=10
*p=10
&a=6487580
p =6487580
&p=6487568
```



```
int a;
double b;
int *p= &b;//wrong//
int *q = &a;//correct
double *r =p;//wrong.
```

```
int x=5;
int *p=&x;
int *pp;
pp=x;
printf("%dp=", p);
printf("%d pp=", pp);
```

Pointer to pointer :

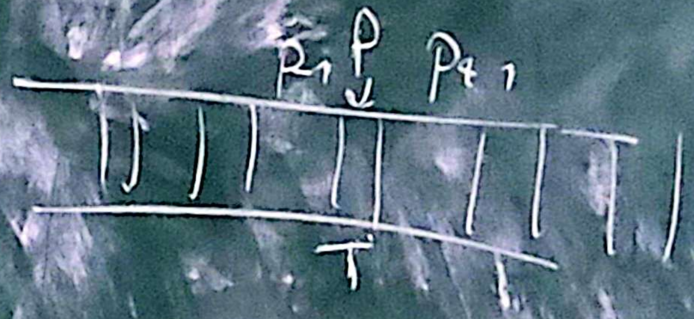
```
#include<stdio.h>
int main(void )
int a=10;
int *pa= &a;
int **pp = &pa;
printf(" the value of a before modification a= %d", a);
*pa=20;
printf("a =%d", a);
printf("*pa=%d", pa);
printf("pa=%d", pa );
printf("**pp=%d", **pp);
printf(" * pp =%d", *pp);
printf( "pp= %d", pp);
return0; }.
```

```
a=10
```

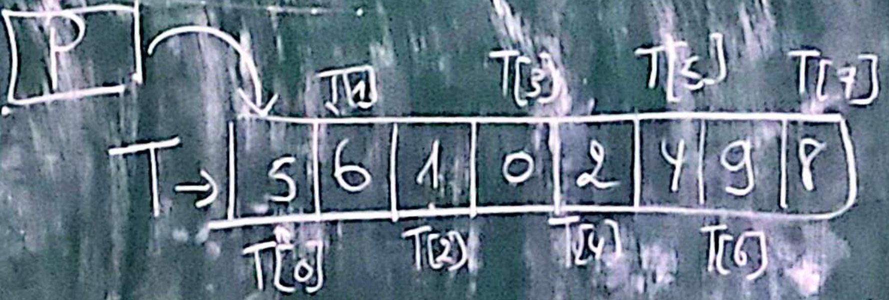
```
a = 20
*pa = 20
pa = 6487572 (l'adresse de a=20)
**pp = 20;
*pp = 6487572
pp = 6487560
```

Module: Algorithms & Data Structures

Course: Pointers ptr=@



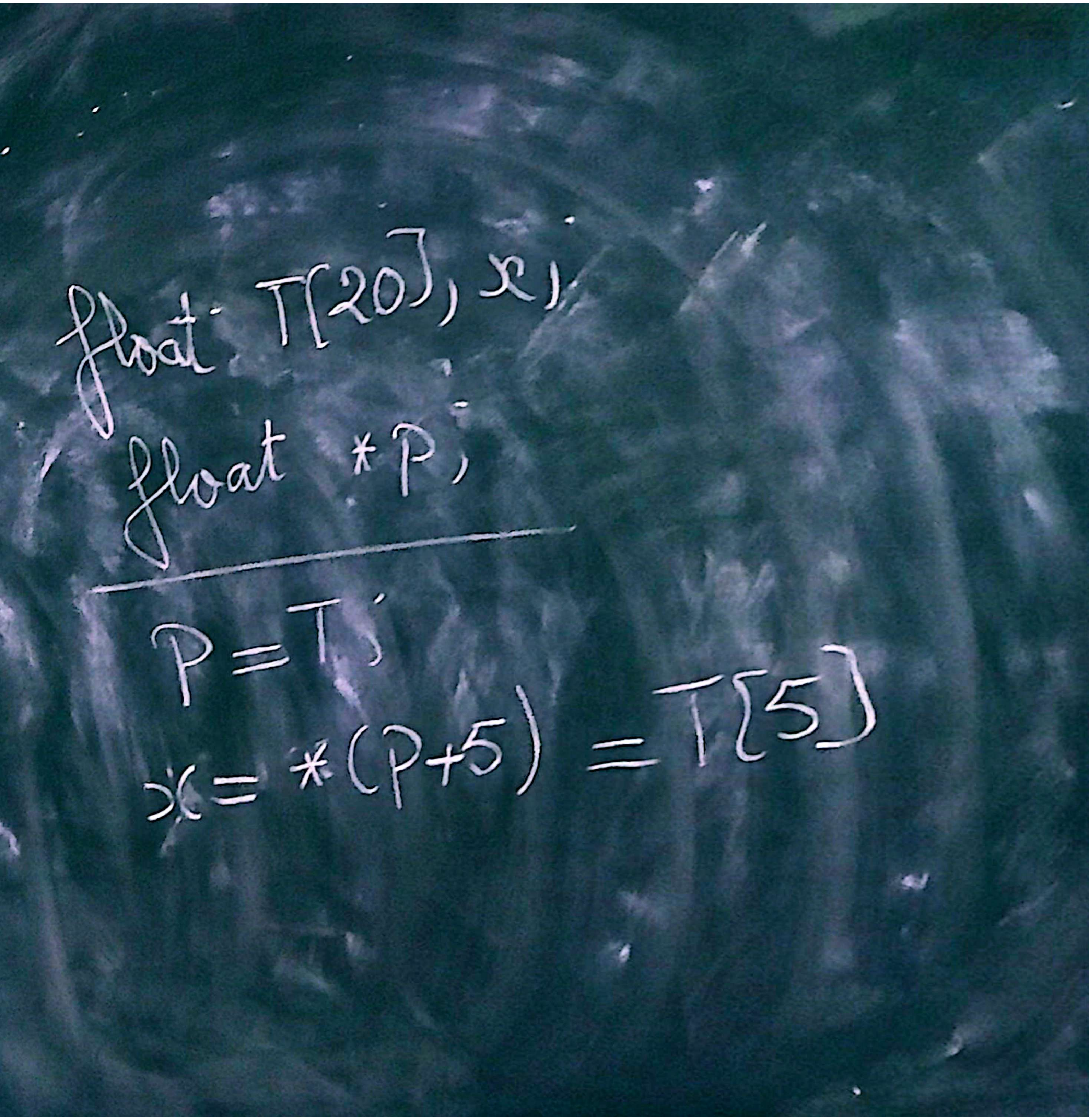
Pointers & Arrays:



```
int T[10];
int *P = T;
P = T;
*(P+1);
```

```
*P = 5
*(P+3) = T[3] = 0
*(P+1) = T[1] = 6
```

```
P+ = n ⟺ P=P+n;
P++ ⟺ P=P+1;
P-- ⟺ P=P-1;
P-=n ⟺ P=P-n;
```



```
float T[20], x,
float *P;
P = T;
x = *(P+5) = T[5]
```

Substruction & comparison of 2 pointers