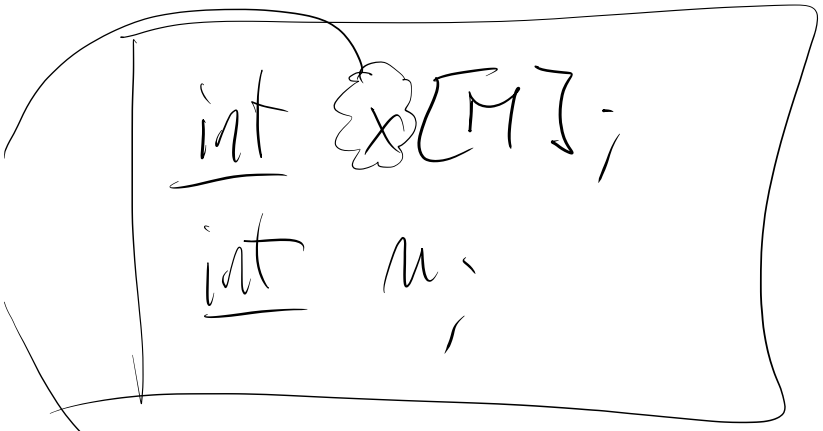
ARITMETICA DEI PUNTATORI

var x [5]int | int x[5];



x[7]

x[0]

↳ rappresenta una locazione di memoria

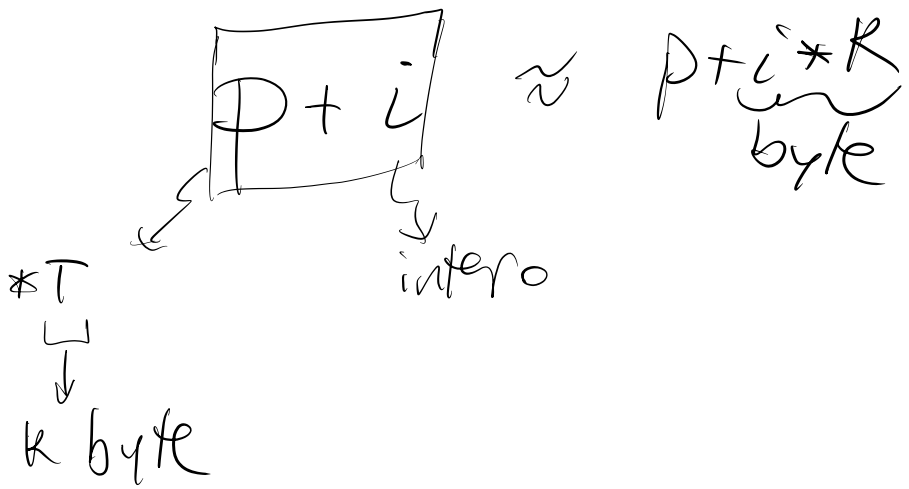
$x \equiv \&(x[0])$

```
int sum(int x[], int n)

int sum(int *x, int n) {
    int sum, i;
    sum = 0;
    for (i=0; i<n; i++)
        sum += x[i];
    return sum;
}

int main() {
    int a[100];
    ...
    printf("%d", sum(a, 20));
}
```

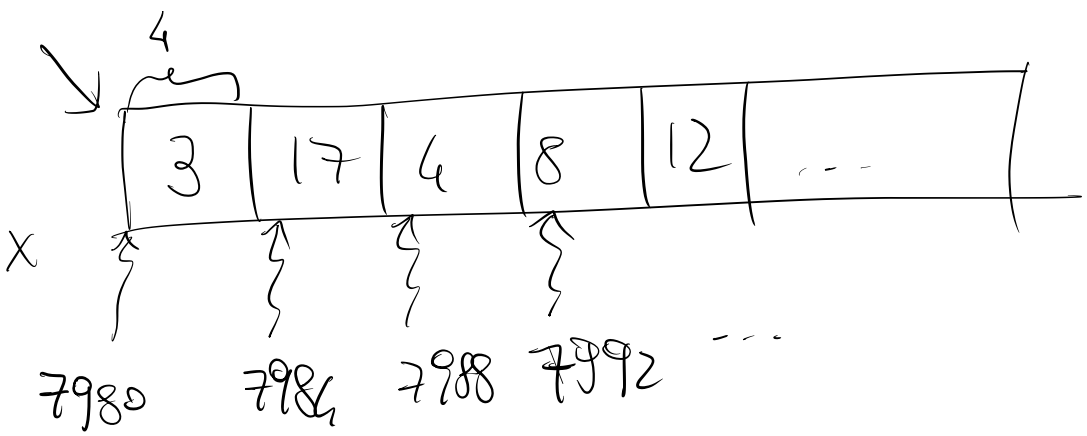
x[i] ↔ *(x+i)

OPERAZIONI CON PUNTATORI

- PUNTATORE + INTERO → PUNTATORE
- PUNTATORE − PUNTATORE → INTERO



```
int sum (int *x, int n) {
    int sum;
    sum = 0;
    int *curs;
    for (curs = x; curs - x < n; curs++)
        sum += *curs;

    return sum;
}
```



ALLOCAZIONE DINAMICA

```
var p *int
p = new (int)
*p = 7
```

```
#import <stdlib.h>
int *p;
p = (int*) malloc(sizeof(int))
*p = 7
```

```
int *p;
p = (int*) malloc(100 * sizeof(int))
for(i=0; i<100; i++)
    p[i] = 7;
...
free(p);
```

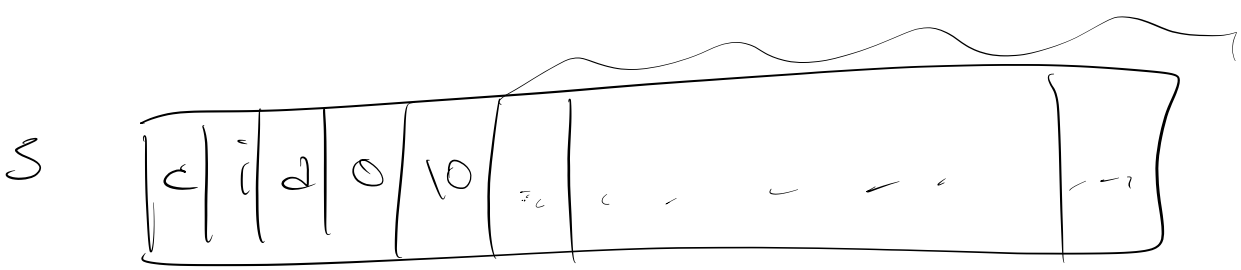
↝ MEMORY LEAK

STRINGHE IN C

= array di char terminati da 0

```
char s[100];
s[0] = 'c';
s[1] = 'i';
s[2] = '2';
s[3] = '0';
s[4] = 0;
```

`s[4] = '\0';`




```
printf("%s\n", s);
```