tipo degli array e aritmetica dei puntatori

5.3 e 5.4 del testo

che tipo ha un array?

double X[20];

X ha tipo double * o double[]

NON double[20]

X è costante

X = 0 o anche X = X+1 ERRORE

double X[20]; X punta al primo elemento X[0]
X ha R-valore &X[0]

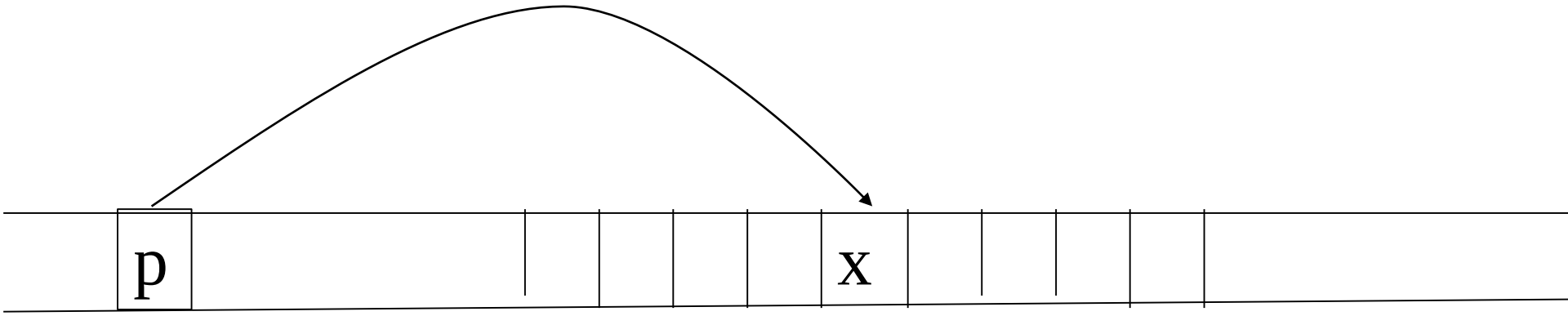
X[0] è il primo elemento di X
X[1] è il secondo eccetera

ma anche
*X
*(X+1)
*(X+2)

* e +   oppure [ ] sono equivalenti

in C++ array sono puntatori, ma anche puntatori sono array

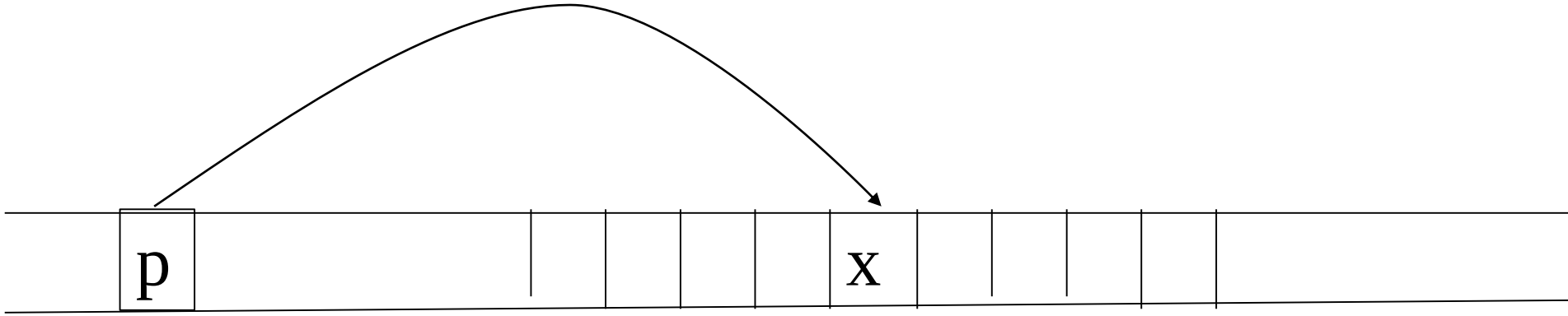
double x, *p=&x;



è come se x fosse l'elemento di un array
quindi x[0], x[1], x[2], ... o *x, *(x+1), *(x+2) ...

il programmatore deve sapere cosa fa ...

questa è l'idea dell'aritmetica dei puntatori



...p-2, p-1, p, p+1, p+2,... sposta il puntatore p sui diversi elementi dell'array (immaginario)

p - 2 = p - 2 * sizeof(tipo elemento puntato)
p + 2 = p + 2* sizeof(tipo elemento puntato)

quindi p + 2 ha valori diversi a seconda di cosa punta p

se
int * p; allora p + 2 = p + 2 * 4
double * p; p + 2 = p + 2 * 8

char * p; p + 2 = p + 2

tutto semplice ? Si, ma ci sono gli array a più dimensioni

che tipo hanno?

che tipo ha un array ?

int A[100] -> int* o int[]
int X[5][10] -> int (*) [10] o int [][10]
int Y[5][5][10] -> int (*) [5][10] o int [][5][10]
int Z[6][5][5][10] -> int (*)[5][5][10] o
int [][5][5][10]

il loro tipo ci dice la dimensione dell'oggetto puntato:
per Z è 5*5*10*4 (byte)

guardiamo bene

double F[3][5][7][9];
tipo di F = double (*)[5][7][9]
R-valore di F = &F[0][0][0][0]

se dereferenziamo un puntatore otteniamo l'oggetto puntato.
In questo caso, *F= double [5][7][9] che ha tipo double (*) [7][9], quindi: il tipo di *F è:
double (*)[7][9]

ricorda che *F=F[0]

ma qual è l'R-valore di *F?

sempre &F[0][0][0][0]

conviene fare una figura: l'array F è una sequenza di 3 torte ognuna formata da 5 strati con 7 righe e 9 colonne.
F punta alla prima torta,
*F punta al primo strato della prima torta
**F punta alla prima riga del primo strato della prima torta
***F punta al primo elemento della prima riga...
****F è il primo elemento della prima riga...

scrivendo i tipi :

*F ha tipo double (*) [7][9]

**F ha tipo double (*)[9]

***F ha tipo double *

****F ha tipo double

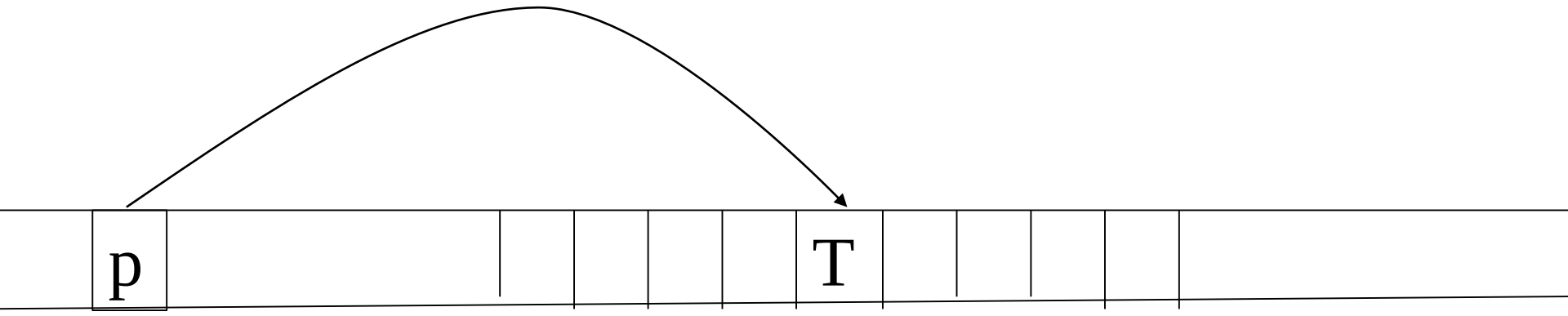
ma tutti hanno lo stesso valore = &F[0][0][0][0]

cout<<F<<*F<<**F<<***F ;

stampa 4 volte lo stesso indirizzo che è l'L-valore del primo elemento dell'array

mettiamo questi tipi degli array assieme all'aritmetica dei puntatori

idea dell'aritmetica dei puntatori



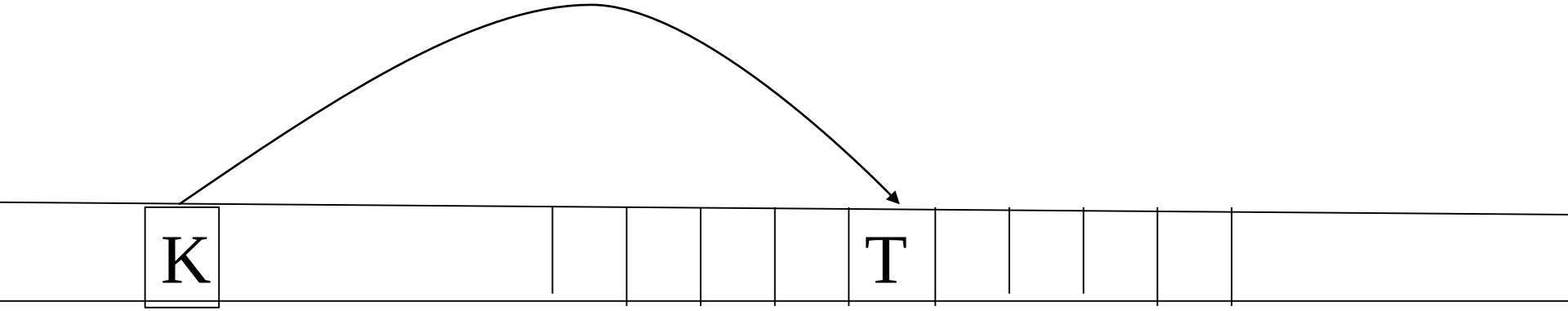
p punta ad un elemento di un array di elementi di tipo T, quindi p+n sposta il puntatore di n elementi a destra e p-n lo sposta di n elementi a sinistra

esempi con array

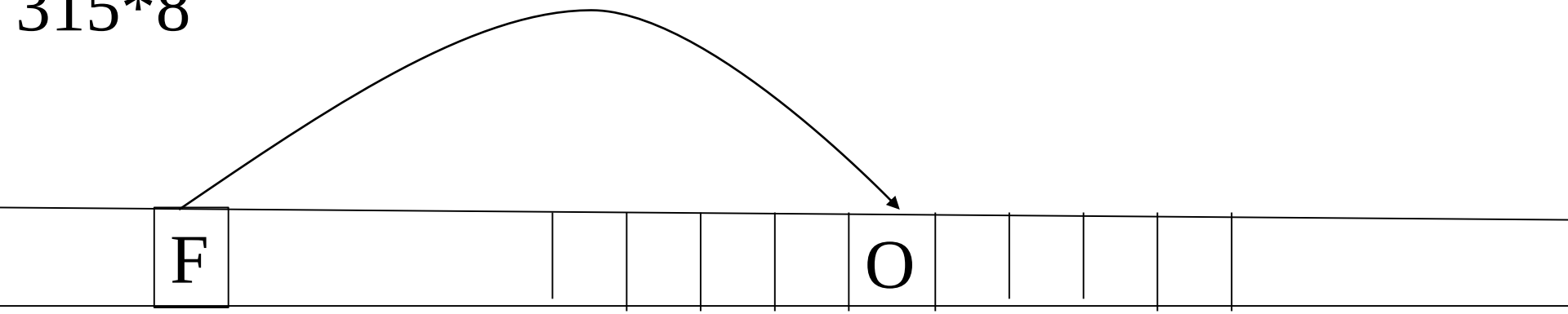
int K[5][10]; tipo di K = int (*) [10]
dimensione=10*4

char K[4][6][8]; tipo = char (*) [6][8] dimensione=6*8

double K[3][5][7][9]; tipo = double (*)[5][7][9]
dimensione=5*7*9*8



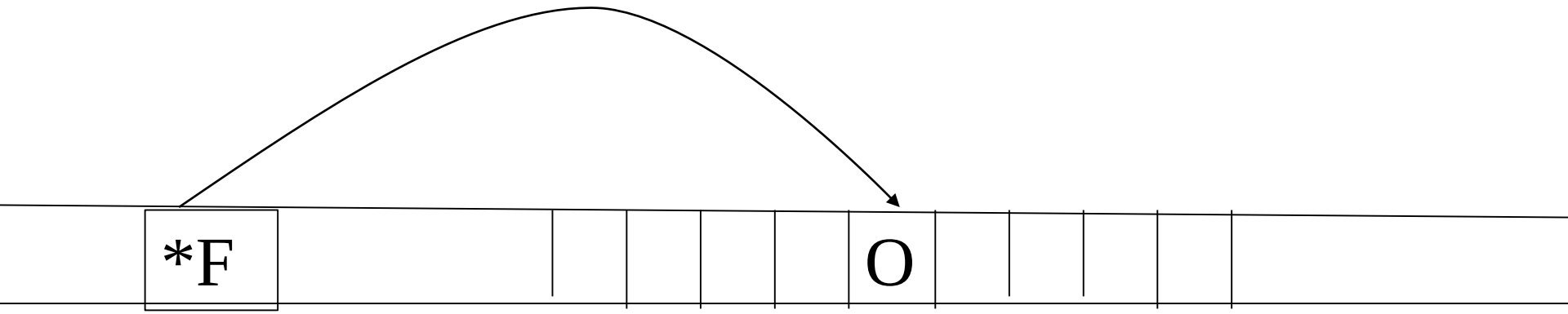
partiamo da F, ha tipo, double (*) [5][7][9]
punta ad un oggetto O di dimensione = (5*7*9)*8 = 315*8



F+1 = F + (315*8)
F+2=F + (2*315*8)
F-5 =F - (5*315*8)
eccetera

sono tutti valori di tipo double (*)[5][7][9]

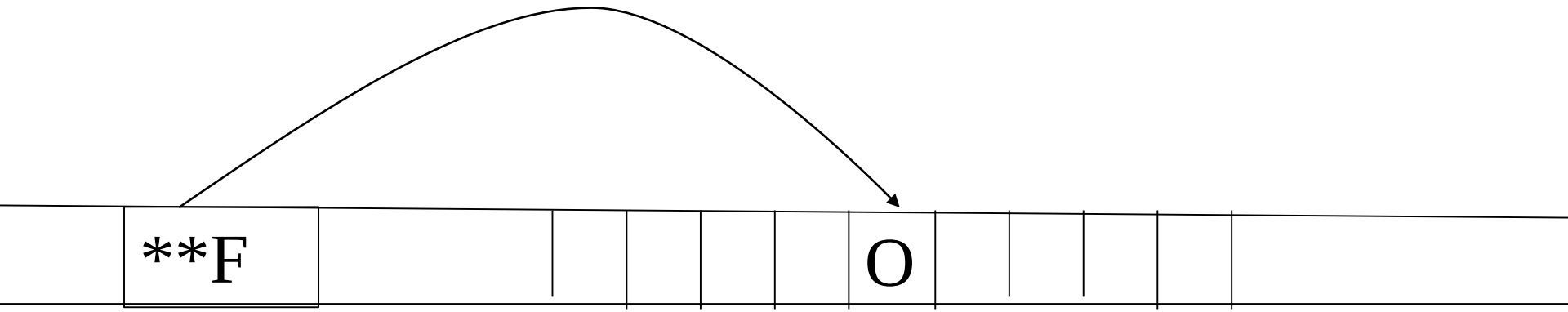
*F ha tipo double (*) [7][9]
punta d un oggetto O di dimensione = (7*9)*8=63*8



*F +1 = F+63*8
*F+2=F+2*63*8
*F-5 =F-5*63*8
eccetera

tutti valori di tipo
double (*)[7][9]

**F ha tipo double (*) [9]
punta ad un oggetto O di dimensione = 9*8=72



**F +1 = F+72
**F+2=F+2*72
**F-5 =F-5*72
eccetera

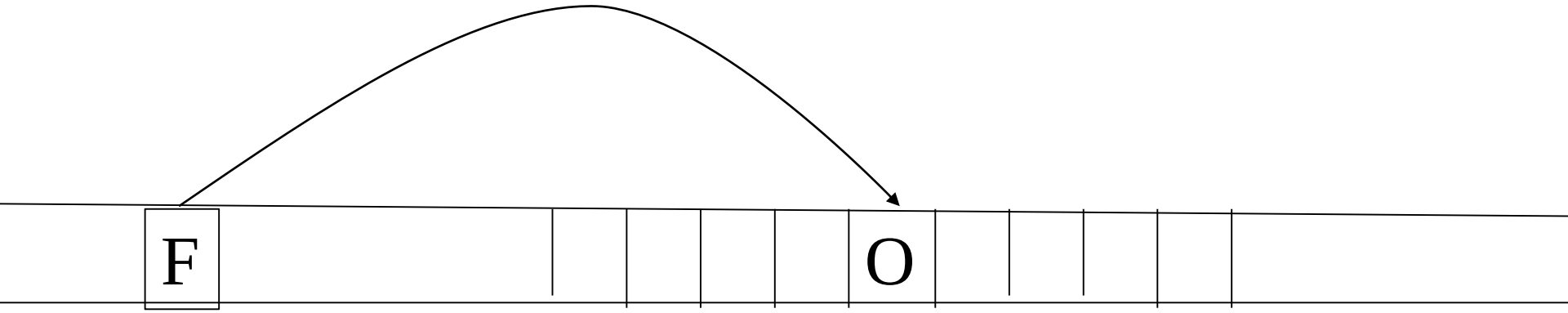
tutti valori di tipo
double (*)[9]

possiamo usare subscripting ?

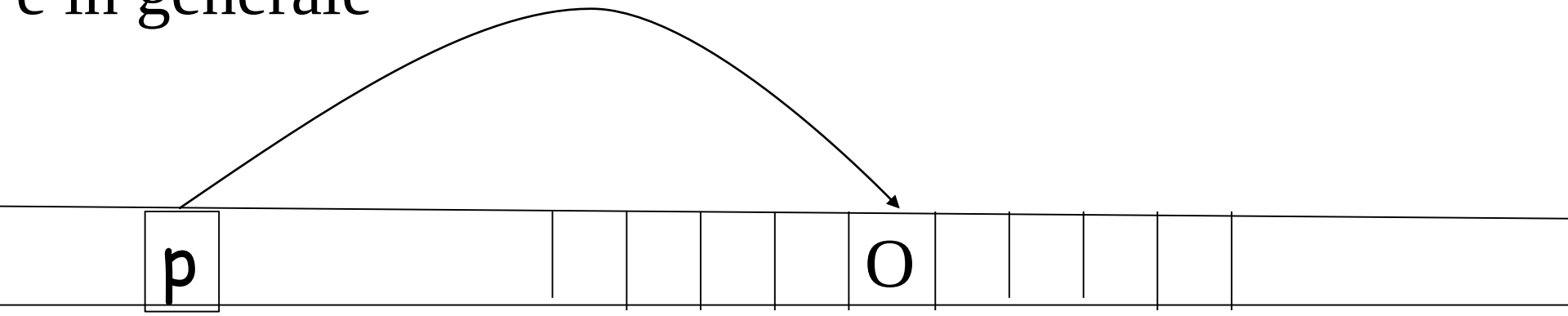
double F[3][5][7][9]; tipo = double (*)[5][7][9]

F[3] = *(F+3) F[-2]=*(F-2) c’è sempre * !!

O ha dimensione = (5*7*9)*8=315*8



e in generale



p[k] =*(p+k)

derefenziazione cambia il tipo
+ cambia il valore

CAPIRE :

float K[3][5][7][10];

K[1]= *(K+1)

K[3][2]=*(*(K+3)+2)

K[2][1][4][1]= *(*(*(*(K+2)+1)+4)+1)

float K[3][5][7][10];

K[-1][-2] = ?

K[-1]= K - (5*7*10)*4 = L1

K[-1][-2] = L1 - 2*(7*10)*4

K[-1][-2][5] = K -(5*7*10)*4 - 2*(7*10)*4 + 5*10*4

esercizi su letsfeedback.com
login = minhw