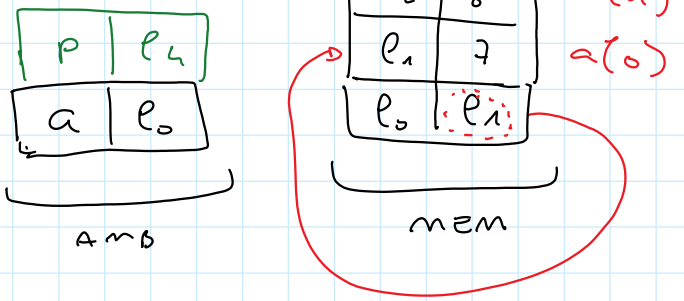
→ ABBIAMO VISTO CHE GLI ARRAY SONO (IMPLICITAMENTE) DEI PUNTATORI

```
int a[3];
```



POSSIAMO ASSEGNARE UN ARRAY A UN PUNTATORE

```
int a[10];
int *p;
  ⋮
p = a;
```

⇐ CORRISPONDE A FARE

```
p = &a[0]
```

LA DIFFERENZA TRA UN ARRAY E UNA VARIABILE PUNTATORE È CHE L'ARRAY NON LO POSSO ASSEGNARE

~~a = p~~

~~&a~~

→ VEDIAMO MEGLIO COSA SIGNIFICA a[i]

↳ USIAMO L'ARITMETICA DEI PUNTATORI

DATA UNA VARIABILE p DI TIPO int *

POSSO SCRIVERE

p + 1

⇗

È UN PUNTATORE CHE PUNTA ALLA LOCAZIONE SUCCESSIVA A QUELLA PUNTATA DA p

(= NON MODIFICA IL VALORE DELLA VARIABILE PUNTATA DA p (≠ *p + 1))

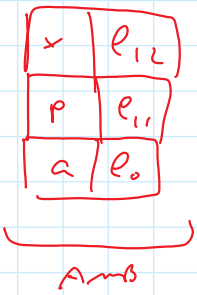
```
int a[10];
int *p;
int x;
  ⋮
p = &a[3];
x = *(p+1)
```

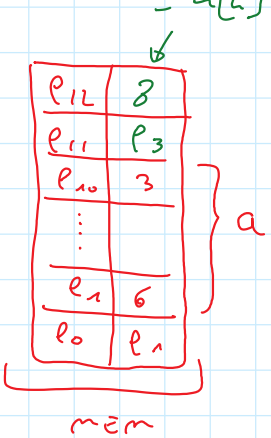
PUNTA ALL'ELEMENTO a[4] ≡ &a[4]

*(&a[4]) ≡ a[4]

CORRISPONDE QUINDI A FARE

x = a[4]





GENERALIZZANDO:

$p + i$ (p PUNTATORE, i INTERO)

È UN PUNTATORE CHE PUNTA i LOCAZIONI DOPO p

$$a[i] \equiv *(a+i)$$

SONO EQUIVALENTI

→ ESEMPIO PROCEDURA SCAMBIA SU ARRAY

(SCAMBIA I VALORI DI DUE ELEMENTI DI UN ARRAY)

```
void scambia (int v[], int dim, int i, int j) {
    int tmp = v[i];
    v[i] = v[j];
    v[j] = tmp;
}
```

POTREI CONTROLLARE CON UN IF CHE i, j < dim

!!! CORRISPONDE A

```
void scambia (int *v, int dim, int i, int j) {
    int tmp = *(v+i);
    *(v+i) = *(v+j);
    *(v+j) = tmp;
}
```

POSSO PASSARE UN ARRAY A UNA PROCEDURA CHE SI ASPETTA UN PUNTATORE (NON IL VICEVERSA) COME PER ASSEGNAMENTI

PROBLEM SOLVING SU ARRAY

→ SCRIVIAMO UNA FUNZIONE CHE CERCA UN ELEMENTO IN UN ARRAY

↳ RESTITUIRE 1 SE L'ELEMENTO È PRESENTE, 0 ALTRIMENTI

RICERCA LINEARE INCERTA

DOVRÒ GUARDARE TUTTI GLI ELEMENTI UNO DOPO L'ALTRO

NON SO SE L'ELEMENTO C'È

L'ELEMENTO CHE STO CERCANDO

```
int member (int a[], int dim, int x) {
    for (int i=0; i<dim; i++)
        if (a[i]==x) return 1;
    return 0;
}
```

SE ARRIVO IN FONDO AL CICLO VUOL DIRE CHE L'ELEMENTO NON C'È QUINDI RESTITUISCO 0

ATTENZIONE:

```
int member (int a[], int dim, int x) {
    for (int i=0; i<dim; i++)
        if (a[i]==x) return 1;
        else return 0;
}
```

SBAGLIATISSIMO

x = 4

8	7	4	2
0	1	2	3

RESTITUISCE 0 DOPO AVER VISTO SOLO IL 1° ELEMENTO

USARE UN RETURN ALL'INTERNO DI UN CICLO FOR È

FORTEMENTE SCONSIGLIATO.

CHI LEGGE IL CICLO FOR POTREBBE NON CAPIRE CHE IL CICLO SI PUÒ INTERROMPERE PRIMA CHE i DIVENTI UGUALE A DIM

SOLUZIONE PIU' ELEGANTE

```
int member (int a[], int dim, int x) {
    int i = 0;
    while (i<dim && a[i]!=x)
        i++;
    if (i==dim) return 0;
    else return 1;
}
```

QUI SONO CHIARAMENTE ESPRESSE ENTRAMBE LE CONDIZIONI DI USCITA DEL CICLO

VUOL DIRE CHE NON HO TROVATO x NELL'ARRAY

DOMANDA: SE x NON È PRESENTE A UN CERTO PUNTO i SARÀ UGUALE A dim.

NELLA VALUTAZIONE DELLA GUARDIA DEL WHILE

a[i] != x

CORRISPONDE IN QUESTO CASO A

a[dim] != 0

SONO USCITO DALL'ARRAY

RISPOSTA: TUTTO OK ... IN C && E' UN AND CONDIZIONALE

Il C valuta && in questo modo:

A && B

① si valuta A. SE A E' FALSA L'INTERA CONDIZIONE E' FALSA

② SE A É VERA si valuta B. SE B E' VERA L'INTERA CONDIZIONE E' VERA, ALTRIMENTI E' FALSA

- SOLUZIONE ANCORA PIU' ELEGANTE
  - USIAMO UNA VARIABILE LOGICA (BOOLEANA)
  - È UNA VARIABILE CHE CORRISPONDE A VERO (true) O FALSO (false)
  - MEMORIZZA IL RISULTATO DI UNA CONDIZIONE

In C LE VARIABILI LOGICHE SONO VARIABILI int

```
int b;
```

b == 0 CORRISPONDE A FALSE

b != 0 " A VERO

TIPICAMENTE USEREMO IL VALORE 1 PER RAPPRESENTARE TRUE (VERO)

SOLUZIONE MIGLIORE

```
int member (int a[], int dim, int x) {
    int i = 0;
    int Trovato = 0;
    while (i<dim && !Trovato) {
        if (a[i]==x)
            Trovato = 1;
        i++;
    }
    if (Trovato) return 1;
    else return 0;
```

VARIABILE LOGICA CHE ESPRIME IL FATTO DI AVER TROVATO X IN A (INIZIALMENTE FALSO)

!Trovato
|||
Trovato == 0

SE TROVATO È VERO
Trovato != 0

PIÙ BREVEMENTE

```
return Trovato;
```

COSE DA NON FARE:

→ MANIPOLARE L'INDICE DI UN CICLO FOR

```
:
for (int i=0; i<dim; i++)
    if (a[i]==x)  i=dim;
```

→ INTERROMPE BRUTALMENTE IL CICLO MODIFICANDO i

ERRORE GRAVE

(SEMBRANO IL PROGRAMMA FUNZIONI)

→ USARE RETURN DENTRO AL CICLO (GIÀ VISTO)

→

```
:
int Trovato=0;
for (int i=0; i<dim; i++)
    if (a[i]==x) Trovato=1;
return Trovato
```

// FUNZIONA, È ELEGANTE MA POCO EFFICIENTE (NON INTERROMPO IL CICLO DOPO AVER TROVATO X, QUINDI PERDO TEMPO INUTILMENTE A GUARDARE GLI ELEMENTI

NON È UN ERRORE GRAVE, MA CERCATE DI

CERCATE DI
EVITARE DI
FAR PERDERE
TEMPO AL PROGRAMMA

(INUTILMENTE A
GUARDARE GLI ELEMENTI
SUCCESSIVI)

→

```
for (int i=0 ; i<dim && !Trovato ; i++)
    if (a[i]==x) Trovato=1;
return Trovato
```

FUNZIONA, E'
POCO ELEGANTE
PERCHE' STO USANDO
FOR PER UNA
ITERAZIONE INDETERMINATA
(MEGLIO IL WHILE)